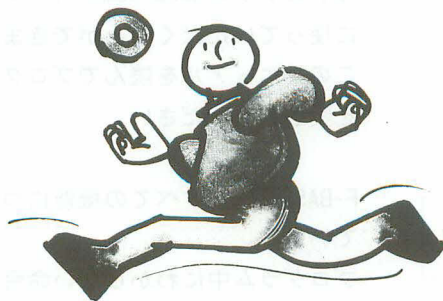
FUJITSU FM SERIES PERSONAL COMPUTER

# F-BASIC386 V2.1ガイド

(F-BASIC386V2.1, F-BASIC386V2.1コンパイラ対応)

FUJITSU

FM TOWNS

FUJITSU FM SERIES PERSONAL COMPUTER

# F-BASIC386 V2・1ガイド

## ごあいさつ

このたびは、F-BASIC386V2.1，F-BASIC386V2.1コンパイラをお買い求めいただき、誠にありがとうございます。F-BASIC386は、本格的32ビットマシンFM TOWNSの各種機能をフルに引き出してプログラムを作ることができる、やさしいプログラミング言語です。グラフィックに、音楽演奏に、音声や効果音の作成に、あるいはビデオ編集や、通信、文字や数値のデータ処理に。32ビットの高速性を生かして、存分にプログラミングを楽しんでいただけることと確信いたしております。

1992年11月

# 本書をお読みになる前に

■FM TOWNS の基本的な操作については、FM TOWNS 本体に添付のマニュアルで説明しています。本書をお読みになる前に必ずご覧ください。

■F-BASIC386に添付のマニュアルについて
F-BASIC386には次の３冊のマニュアルがついています。

| | |
|---|---|
| F-BASIC386V2.1ガイド | F-BASIC386の起動と終了の方法、プログラムの作り方、コンパイラの操作方法、エディタの操作方法などを説明しています。また、盛りだくさんのサンプルプログラムも用意してあります。サンプルプログラムはCD-ROMに入っていますので、すぐに使っていただくことができます。<br>このマニュアルを読んでプログラミングの楽しさを実感してください。 |
| F-BASIC386V2.1リファレンス | F-BASIC386のすべての機能について詳細に解説しています。<br>プログラム中にわからない命令などがあったときにこのマニュアルをお読みください。 |
| F-BASIC386V2.1ポケットブック | F-BASIC386の全命令をコンパクトなポケットブックにまとめました。命令の形式と機能だけの簡単な一覧ですが、必ずお役に立つと思います。プログラミングのときは、ぜひそばに置いてご活用ください。 |

■F-BASIC386には、上記の３冊のマニュアルのほかに、CD-ROMの中にオンラインチュートリアルマニュアル（ベーシックアイランドの冒険）が入っています。
F-BASIC386の入門にお役立てください。

# 表記について

| 本マニュアル内で使用している記号と用語 | このマニュアルでは記述の中に、次のような用語や記号を使っています。それぞれの用語や記号の意味は次のようになります。 |
| --- | --- |
| ご　注　意 | 操作上で特に注意しなければいけないことをまとめてあります。 |
| ひとこと | 操作の参考となるような知識を掲載しています。 |
| 便利な使い方 | F-BASIC386を便利に使うためのノウハウを掲載しています。 |
| チャレンジ | プログラミングに挑戦したり、プログラムを自分で変更して使うためのヒントや方法をまとめてあります。 |
| プログラムの説明 | プログラミングの技法や、処理の内容について説明しています。 |
| 1、2………□ | 操作手順の順番を表しています。 |
| ①、②………○ | ひとつの操作手順の中でさらに操作が必要な場合、その順番を表しています。 |
| ↵キー | リターンキーを表しています。［実行］キーもほぼ同じ働きをします。 |
| □□□キー | キーボード上のキーを表しています。 |
| ［　］ | 画面に表示されるアイコンやボタンを表しています。 |
| ● | 解説の項目単位についています。 |
|  | 操作に対する解説、または本文に対する補足説明であることを表しています。 |
|  | 操作に対する結果であることを表しています。 |
|  | 参照ページを表しています。 |
| カーソル | 画面内の赤い“｜”マークのカーソルです。 |
| マウスカーソル | 画面内の矢印のカーソルです。 |
| 左ボタン | マウスの左側のボタンです。 |
| 右ボタン | マウスの右側のボタンです。 |
| 左クリック | マウスの左ボタンを1回押す操作を言います。 |
| 右クリック | マウスの右ボタンを1回押す操作を言います。 |
| ダブルクリック | マウスの左ボタンを素早く続けて2回押す操作を言います。 |
| ドラッグ | マウスのボタンを押したままでマウスを移動する操作を言います。 |
| 入力 | キーボードやソフトウェアキーボードを使って画面に文字を記入することを言います。 |
| アイコン | 機能や操作などを絵で表現したシンボルマークのことを言います。 |
| ボタン | マウスカーソルを合わせて左クリックすると、対応した機能が使えるアイコンのことを言います。 |
| スクロール | 隠れていて見えない部分を表示するために、表示画面を上下に移動させることを言います。 |

# 目次

| 第１部 | Ｆ－ＢＡＳＩＣ３８６事始め |
| --- | --- |
| | F-BASIC386をはじめて使う方のために、起動と終了の方法を説明します。また、F-BASIC386でプログラムを作るための知識について説明します。そして、その知識を応用するとF-BASIC386ではこんなこともできる、というサンプルプログラムを紹介します。 |

## 序　章　F-BASIC386の世界へようこそ



## 第１章　F-BASIC386を動かしてみよう



## 第２章　基本的なプログラムを作るための知識

## 第３章　本格的なプログラムを作るための知識



## 第４章　F-BASIC386の世界を体験してみよう

| 第２部 | Ｆ－ＢＡＳＩＣ３８６の操作のしかた |
|---|---|
| | プログラムを作るときに便利なエディタの使いかたを説明します。<br>また、プログラムを FM TOWNS が直接実行できる形式に変換するコンパイラの使いかたと、操作上の注意点について説明します。 |

## 第１章　エディタの概要



## 第２章　エディタの操作



## 第３章　コンパイラを使う前に

## 第４章　コンパイラの操作



## 第５章　コンパイラのエラーメッセージ



| 付　録 | このガイドで紹介しているF-BASIC386の機能のまとめや用語の索引を掲載しています。 |
|---|---|

# 第1部

## F-BASIC386の事始め

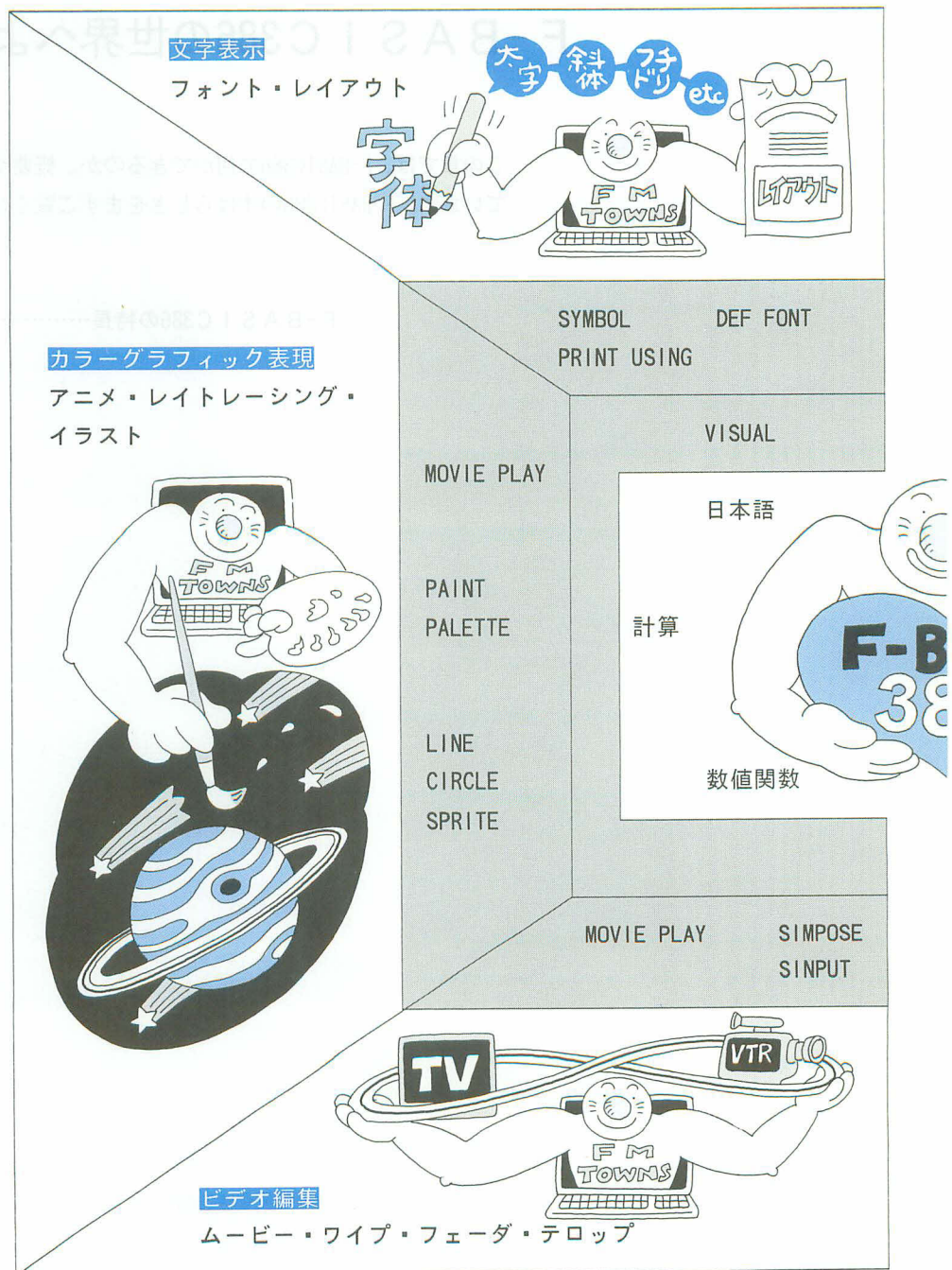
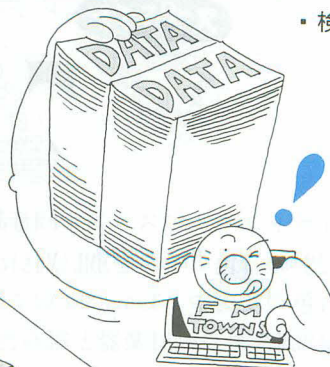
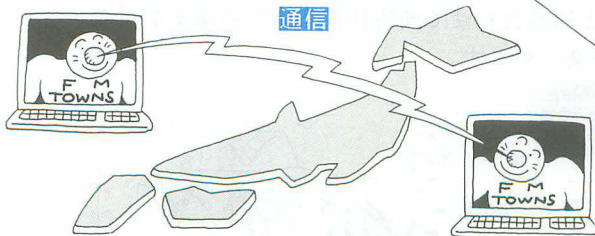
ここでは、F-BASIC386の起動と終了の方法、F-BASIC386でプログラムを作るための知識について説明します。
そしてF-BASIC386でどんなことができるのかを知っていただくために、サンプルプログラムを紹介します。

# 序　章

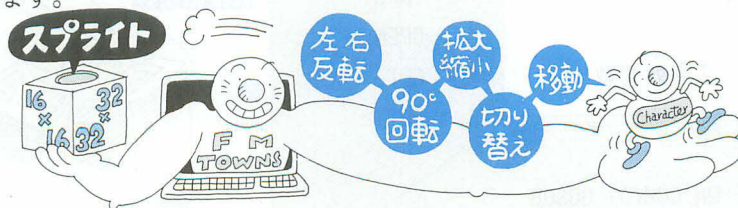
# F-BASIC386の世界へようこそ

この章では、F-BASIC386で何ができるのか、概要や特長を解説しています。F-BASIC386のすばらしさをまずご覧ください。

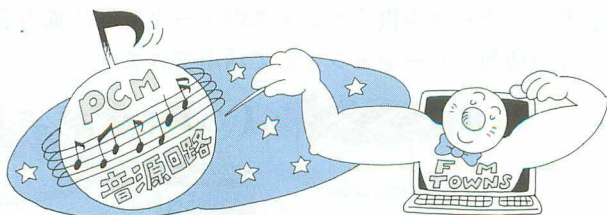
# F－BASIC386の特長

F-BASIC386の
概要図

サンプリング
音声・効果音
ワン
音楽演奏
オーケストラ・MIDI
CD-DA
PCMREC
PCMPLAY
AUDIO
PCM音源
PLAY
CD PLAY
FM音源
INPUT
OPEN
GET
PUT
情報の整理
データの入力・計算
・検索
文字列関数
ON COM(0) GOSUB
OPEN COM(0)
通信
DATA
DATA

## F－BASIC386の特長

| | |
|---|---|
| 多彩な表現ができるグラフィック機能 | ●32,768色、256色、16色と３つのモードのグラフィック画面。<br>●グラフィック画面（32,768色のみ）とビデオ画面の合成。<br>●ビデオ画像をそのままの大きさで扱えるオーバースキャン表示。<br>●グラフィックの縮小、拡大、移動ができるビューポート。<br>●アニメーションデータ（拡張子.MMM）を再生できるアニメーション機能。 |

| | |
|---|---|
| 動くキャラクタが作れるスプライト | ●グラフィック画面、ビデオ画面との合成。<br>●スプライトパターンひとつのサイズは16×16ドット。このパターンを組み合わせて最大32×32パターンのスプライトキャラクタを作ることができます。<br>●使用できる色は32,768色と16色。<br>●左右反転、90°回転、縮小、拡大、切り替え、移動が可能。<br>☞スプライトキャラクタを動かしながら、PCM音源で爆発音を出したり、FM音源で音楽を演奏したり、音と動きをミックスしたゲームやグラフィックスが楽しめます。 |

| | |
|---|---|
| シンセサイザー並みの音楽演奏ができるFM音源 | ●６チャンネルのステレオFM音源回路。<br>●128 の音色データをMML(Music Macro Language) で自在に演奏。<br>●音色の変更や、TownsSOUNDで作成した音色の読み込みが可能。<br>☞PCM音源やMIDI楽器と組み合わせて同時16和音までの演奏が可能。シンセサイザー並みの演奏を楽しめます。 |

## 多彩な音作りが楽しめるPCM音源

- ８チャンネルのステレオPCM音源回路。
- CD-ROM内のリアルなPCM音色をMMLで自在に演奏。
- 音色の変更や、TownsSOUNDで作成した音色データの読み込みが可能。
- 本体に内蔵のマイクを使って音のサンプリングが可能。

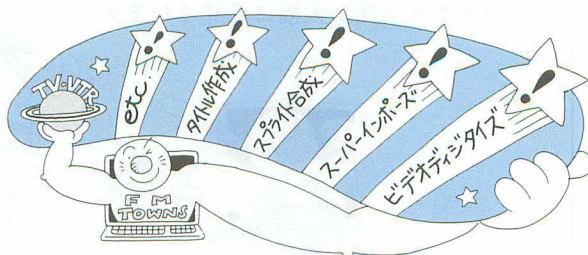
人の声や擬音、効果音など自由な音作りが楽しめます。

## CD-ROMドライブをコントロールできるCD命令

- ７つの命令でCD-ROMドライブをCDプレーヤとして自在にコントロール。

F-BASIC386でCD演奏のプログラムを作ることができます。

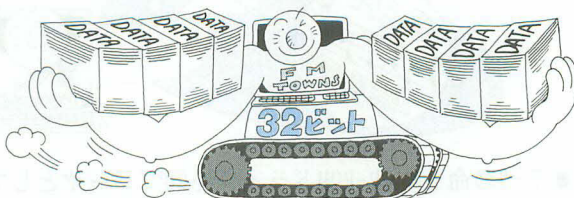
## 本格的な編集ができるビデオ編集機能

- オプションのビデオカードで、テレビやビデオの映像をグラフィック画面に取り込みが可能。
- ビデオやテレビから映像を取り込んだムービーデータを再現するムービー機能。再生する位置や速度は自由自在。
- ビデオ画面にグラフィック画面をスーパーインポーズ。
- ビデオ画面にスプライトを合成。

ビデオカードの出力をお手持ちのビデオデッキで録画して、ホームビデオのタイトルやスーパーインポーズの作成ができます。

## F－BASIC386の特長

**情報処理に威力を発揮する32ビットCPU**

- 32ビットCPU のネイティブモードならではの広大なアドレス空間。
- －2147483648 ～ ＋2147483647 までの数値が扱えるロング型整数による高速演算。
- さらに演算速度を高速にする数値演算プロセッサ（オプション）。
- データ通信対応のRS-232C 通信ポートやオプションのモデムカード。

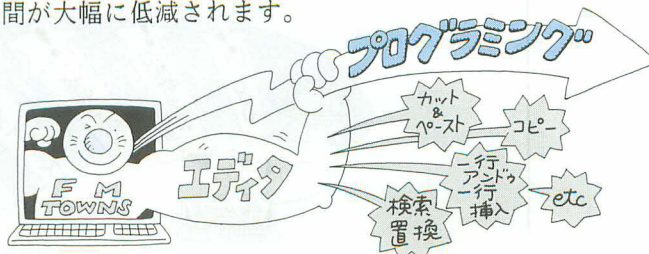
多くのデータを扱うビジネスのデータ処理に威力を発揮。FMTOWNS をビジネスの情報ステーションとして活用できます。

**プログラミングの効率を高めるエディタとインタプリタ**

- プログラムを編集する高機能フルスクリーンエディタ。
- プログラムを実行するインタプリタ。

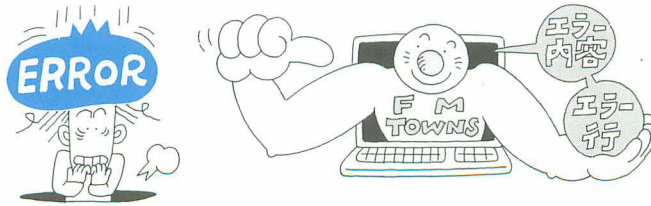
プログラムを実行してもリストが消されたりすることはありません。

**プログラミングの能率を上げるエディタの機能**

- すべての機能がワンタッチで使えるマウス対応のプルダウンメニュー。
- プログラムの編集を容易にする一行UNDO、一行挿入、検索、置換、カット、コピー、ペースト。
- デバッグに便利な部分実行(Run行番号)、一行実行、トレース(Run(Tron))機能。
- 一度保存をおこなうとファイル名を覚えているファイルウィンドウ。

何度も同じプログラム文を書く手間や、プログラム文を探したり、入れ替えたりする手間が大幅に低減されます。

**デバッグに便利なエラー処理機能**

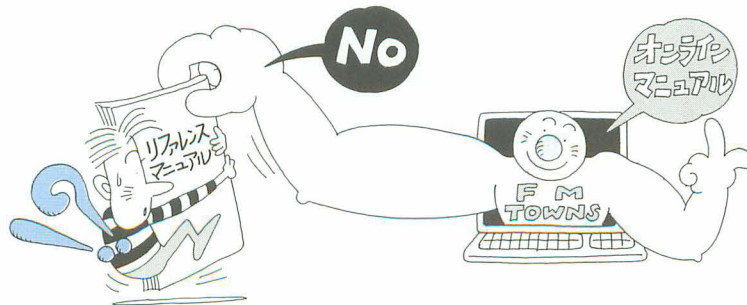
- ガイドウィンドウによるエラー内容表示。
- エディタのエラー行表示。
- ヘルプ機能によるエラーの説明画面呼び出し。

☝エラー内容の確認やデバッグに非常に便利です。

**困ったときに役に立つオンラインマニュアル（F-BASIC386M ）**

- 知りたい用語でオンラインマニュアルを直接さがせるキーワード検索。命令を覚えていなくても検索できます。
- キーワードに該当する解説がなくてもさがせる索引表示。
- 使い勝手を考慮した「読み」、「さっきのページ」、「エラー番号」、「しおり」での検索。

☝プログラミング中に使い方や構文がわからなくなったときに、リファレンスマニュアルをさがさなくても、画面で解説や構文を見ることができます。

☝オンラインマニュアルはF-BASIC386Mを起動したときに使えます。

# 第1章

# F-BASIC386を動かしてみよう

この章では、F-BASIC386の起動と終了の方法を、CD-ROMの場合とハードディスクの場合に分けて解説しています。あなたのシステムに合わせて必要な方をご覧ください。
また、CD-ROMの内容と、楽しい絵と音楽でF-BASIC386のプログラミングの基本が学べる「ベーシックアイ ランドの冒険」の使い方を解説しています。

# 1. 起動の方法（CD−ROM）

**CD-ROMからの起動**

F-BASIC386をCD-ROMから起動するときは、必ず次のような方法で起動してください。

他のアプリケーションのTownsMENU からは起動しないでください。

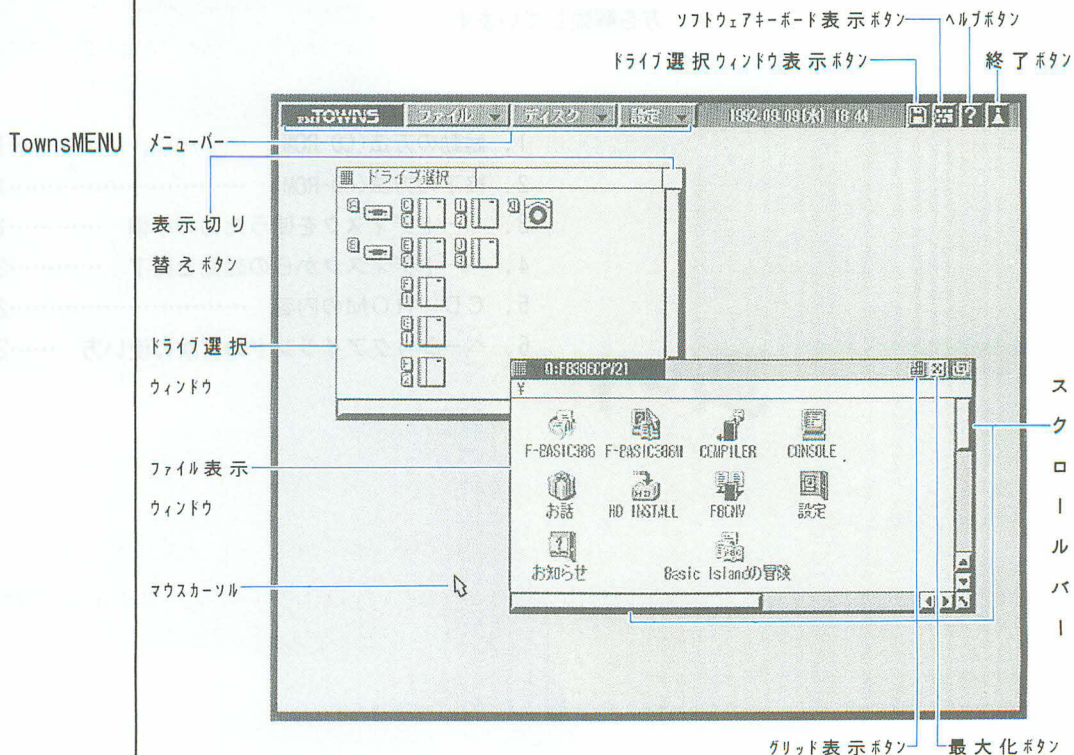
**1 電源を入れる**

本体前面の電源スイッチを押し、電源を入れます。

「システムをセットしてください」とメッセージが表示されます。

**2 CD-ROMをセットする**

F-BASIC386のCD-ROMをセットします。

CD-ROMが回り始め、しばらくすると「TownsMENU」が表示されます。

TownsMENU

**ひとこと**

● すでにハードディスクに Townsシステムソフトウェアなどが複写されているときや、他のCD-ROMやフロッピィディスクがセットされているときは、他のTowns MENUが表示されます。F-BASIC386のCD-ROMをセットして起動ドライブをCD-ROMにした後、いったん電源を切って、もう一度電源を入れ直してください。

# 1. 起動の方法（CD－ROM）

**3** F-BASIC386を選択する

マウスカーソルをTownsMENU の[F-BASIC386]あるいは[F-BASIC386M]のアイコンに合わせ、左クリックします。

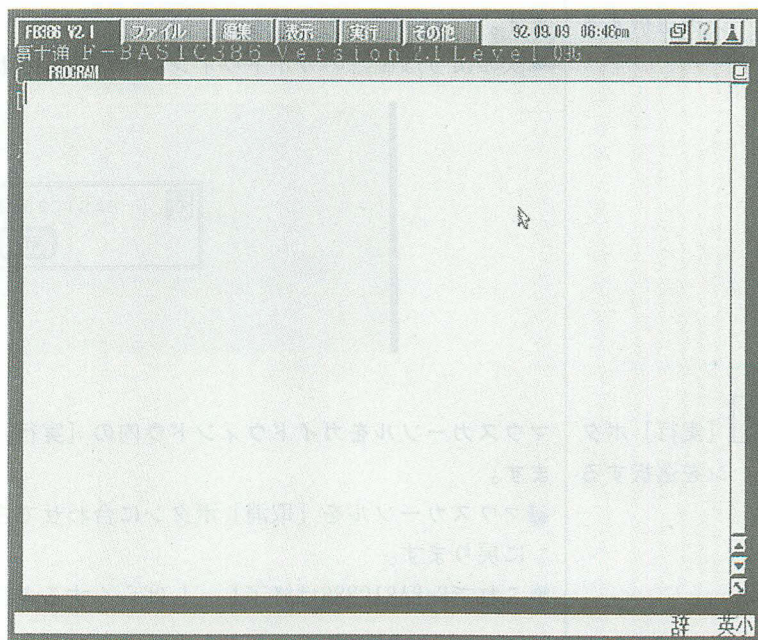
間違って選択したときは、もう一度左クリックすると選択が取り消されます。

**4** F-BASIC386を実行する

**マウスカーソルをメニューバー［ファイル］のサブメニュー［実行／開く］に合わせ、左クリックします。**

メニューバー［ファイル］を左クリックするとサブメニューが表示されます。

しばらくすると、F-BASIC386のエディタの画面が表示され、F-BASIC386が使えるようになります。（エディタの画面☞ 170ページ）

エディタの画面

ひとこと

● 手順3で［お知らせ］を選択すると、F-BASIC386を使う上での注意事項や補足説明を見ることができます。

● CD-ROMは、フロッピィディスクのように、万一に備えてバックアップを取る必要はありません。常にCD-ROMをセットしてF-BASIC386を起動してください。

ご注意

● F-BASIC386M ではオンラインマニュアルを参照しながらプログラミングができますが、F-BASIC386より使えるプログラムメモリの容量が小さくなります。

# 2. 終了の方法（CD−ROM）

**終了の方法**

F-BASIC386は、次のような方法で終了してください。

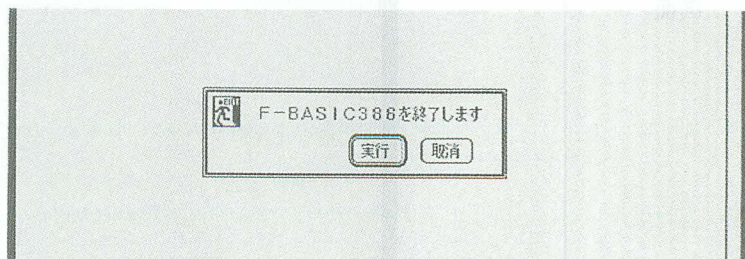
エディタの画面で、ファイルの読み込みなどの操作をおこなっているときは、操作を終了させてください。

**ご注意**

● 終了の操作をおこなうとエディタ内に表示されていたプログラムは消去されます。必要なプログラムは忘れずにフロッピィディスクやハードディスクに保存しておいてください。（プログラムの保存☞ 211ページ）

**1 ［終了］ボタンを選択する**

**マウスカーソルをエディタの画面右上の［終了］ボタンに合わせ、左クリックします。**

次のような確認のガイドウィンドウが表示されます。

**2 ［実行］ボタンを選択する**

**マウスカーソルをガイドウィンドウ内の［実行］ボタンに合わせ、左クリックします。**

マウスカーソルを［取消］ボタンに合わせて左クリックするとエディタの画面に戻ります。

これでF-BASIC386は終了し、しばらくするとTownsMENUに戻ります。

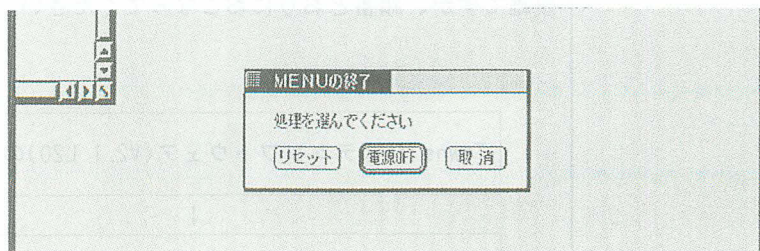
**3 CD-ROMを取り出す**

**CD-ROMドライブから、F-BASIC386のCD-ROMを取り出します。**

①OPENボタンを押し、CD-ROMドライブを開けます。
②F-BASIC386のCD-ROMを取り出し、ケースに戻します。
③CD-ROMドライブを閉じます。

**4** ［終了］ボタンを選択する

マウスカーソルをTownsMENU 画面右上の［終了］ボタンに合わせ、左クリックします。

次のようなガイドウィンドウが表示されます。

**5** ［電源OFF ］を選択する

マウスカーソルを［電源OFF ］に合わせ、左クリックします。

FMTOWNS の電源が切られます。

マウスカーソルを［取消］ボタンに合わせて左クリックすると、TownsMENU に戻ります。

**ひとこと**

- 上記の操作のかわりにエディタ画面でSYSTEM命令を実行しても、F-BASIC386は終了し、TownsMENU に戻ります。
  SYSTEM命令の使い方については、「F-BASIC386 V2.1 リファレンス」をご覧ください。
- 上記操作の手順5で［リセット］を選択すると、電源スイッチを切らずにFMTOWNS を起動し直すことができます。

**ご注意**

- 取り出したF-BASIC386のCD-ROMは、必ずケースに入れて保管してください。データ記録面にキズがついたり、ゆがんだりすると使えなくなってしまいます。

# 3. ハードディスクを使うための準備

**F-BASIC386をハードディスクから起動するための準備**

ハードディスクが使えるFMTOWNSのシステムでは、F-BASIC386のシステムをハードディスクへ複写して使うことができます。ハードディスクへ複写することを、インストールといいます。ただし、最初に次のような準備作業が必要です。少し複雑ですが、順番どおりにおこなってください。

Townsシステムソフトウェア(V2.1 L20)のインストール　　すでに複写されているときは不要

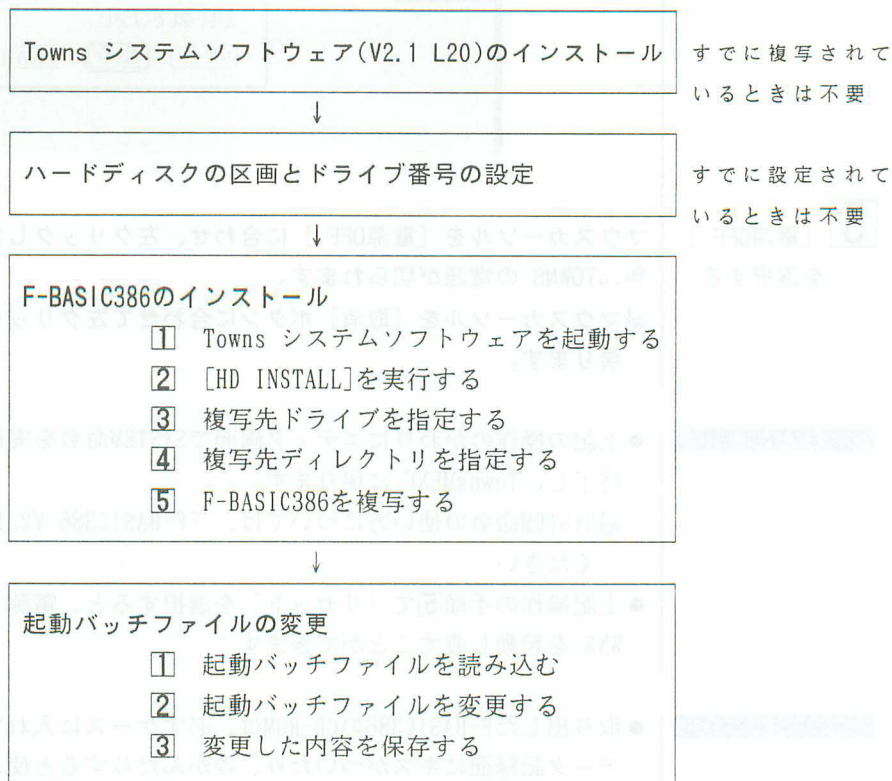
↓

ハードディスクの区画とドライブ番号の設定　　すでに設定されているときは不要

↓

F-BASIC386のインストール
1. Townsシステムソフトウェアを起動する
2. [HD INSTALL]を実行する
3. 複写先ドライブを指定する
4. 複写先ディレクトリを指定する
5. F-BASIC386を複写する

↓

起動バッチファイルの変更
1. 起動バッチファイルを読み込む
2. 起動バッチファイルを変更する
3. 変更した内容を保存する

この作業は最初に一度おこなえば、何らかの理由でハードディスクのF-BASIC386のシステムを消してしまわない限り、再度おこなう必要はありません。

**ご注意**

● F-BASIC386をインストールするには、ハードディスクに10MB以上の容量が必要です。

## 3. ハードディスクを使うための準備

3

### Townsシステムソフトウェア(V2.1 L20)のインストール

Townsシステムソフトウェア(V2.1 L20)を、ハードディスクへインストールします。

- すでにハードディスクにTownsシステムソフトウェア(V2.1 L20)がインストールしてある場合は、この操作は必要ありません。
- F-BASIC386は、旧Townsシステムソフトウェア(V1.1 L10 /L20 /L30)から起動して使うことはできません。もし、ハードディスクに旧Townsシステムソフトウェアが入っているときは、必ずTownsシステムソフトウェア(V2.1 L20)に入れ換えてください。
- システムのインストールは、Townsシステムソフトウェアの［ツール］の［HD INSTALL］でおこないます。具体的な操作方法についてはTownsシステムソフトウェアに添付のマニュアルをご覧ください。

### ハードディスクの区画とドライブ番号の設定

F-BASIC386をインストールするための区画とドライブ番号を設定します。

- すでにハードディスクにF-BASIC386用の区画とドライブ番号が設定されているときは、この操作は必要ありません。
- Townsシステムソフトウェアをインストールした区画に10MB以上の容量が残っていれば、同じドライブにインストールしてもかまいません。ただし、管理のしやすさという点で、F-BASIC386用に新しく区画とドライブを設定することをおすすめします。
- 作業はメニューバー［設定］のサブメニュー［区画設定］と［ドライブ構成］でおこないます。具体的な操作方法についてはTownsシステムソフトウェアに添付のマニュアルをご覧ください。
- 設定した内容は、FMTOWNS を起動し直さないと有効になりません。

**ひとこと**

● 区画とは、1台のハードディスク内をいくつかの独立した部分に分割し、分割されたそれぞれの部分のことを言います。区画に分割すると、それぞれを1つのドライブとして扱うことができるようになり、大容量のハードディスクの取り扱いが非常に楽になります。

## 3. ハードディスクを使うための準備

**F-BASIC386のインストール**

F-BASIC386をハードディスクへインストールします。

**1 Townsシステムソフトウェアを起動する**

Townsシステムソフトウェアをハードディスクから起動します。

CD-ROMやフロッピィディスクを取り出して起動すると、ハードディスクから起動することができます。

**2 [HD INSTALL]を実行する**

①F-BASIC386のCD-ROMをセットする

②F-BASIC386のTownsMENU を表示する

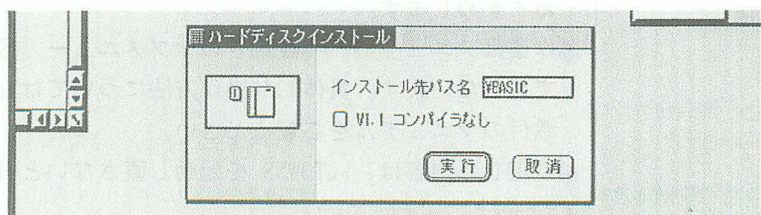
ドライブ選択ウィンドウでQドライブを左クリックします。

メニューバー［ファイル］のサブメニューで［実行／開く］を左クリックします。

③［HD INSTALL］を実行する

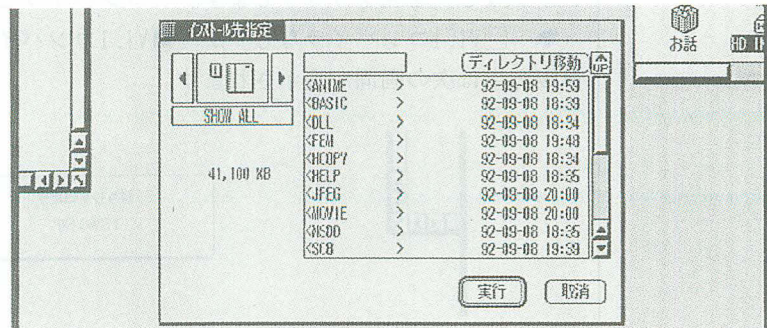
［HD INSTALL］のアイコンを左クリックします。

メニューバー［ファイル］のサブメニューで［実行／開く］を左クリックします。

次のようなガイドウィンドウが表示されます。

何も変更せずに［実行］を左クリックすると、自動的にDドライブのルートディレクトリに<BASIC> というサブディレクトリが作られます。

**3 複写先ドライブを指定する**

①ハードディスクのアイコンを左クリックします。

次のようなガイドウィンドウが表示されます。

②◀または▶をクリックして複写先のドライブを表示します。

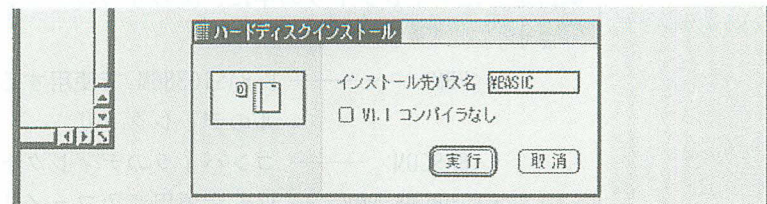
［SHOW ALL］でドライブを指定することもできます。

**4 複写先ディレクトリを指定する**

①複写先ドライブのディレクトリ一覧の中から複写するディレクトリを左クリックします。

ルートディレクトリに複写することもできますが、管理のしやすさといったことから、なるべくサブディレクトリに複写してください。

②［実行］を左クリックします。

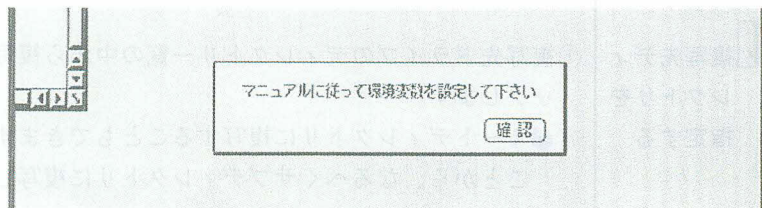
指定したドライブとディレクトリが表示されます。

**5** F-BASIC386を複写する

［実行］を左クリックします。

V2.1のコンパイラの他にV1.1のコンパイラも複写する場合は［□V1.1コンパイラなし］を左クリックします。（コンパイラとは 224ページ）

［□V1.1コンパイラなし］が［■V1.1コンパイラあり］に変わります。

複写中は次の画面が表示されます。

複写が終了すると、次のメッセージが表示されます。
［確認］を左クリックしてください。

TownsMENU に戻ります。

指定したディレクトリの中に、次の６つのディレクトリとファイルが複写されます。

- <FBM> ――――― F-BASIC386M で使用するオンラインマニュアルファイルのディレクトリ
- <BASCOM> ―――― コンパイラのディレクトリ
- E_DRUMS.PMB ― PCM 音源用音色ファイル
- FB386.EXP ―― F-BASIC386のプログラム
- FB386M.EXP ―― F-BASIC386M のプログラム
- FM_1.FMB ―― FM音源用音色ファイル

ひとこと

●ディレクトリについては、☞27ページをご覧ください。
●ルートディレクトリについては、☞26ページをご覧ください。
●サブディレクトリについては、☞26ページをご覧ください。

便利な使い方

F-BASIC386は自動的にアイテム登録されますが、次のようなパラメータを入力してアイテム登録し直すと、F-BASIC386を起動したときにエディタ画面が表示されずに、直接、指定したBASIC のプログラムが実行されます。
●パラメータに次のような方法で実行したいプログラム名を入力します。

パラメータ：　-X A:¥ABCDEFGH.BAS

-X ―――― BASIC プログラムの直接実行の記号、必ずつける
A:¥ ―――― プログラムがあるドライブとディレクトリの指定
ABCDEFGH.BAS ― プログラム名

## 3. ハードディスクを使うための準備

**起動バッチファイルの変更**

起動バッチファイルとは、FMTOWNS を起動するために必要な設定が書き込んであるファイルです。ファイル名は "AUTOEXEC.BAT" と決められています。
この "AUTOEXEC.BAT" を、F-BASIC386の使用環境に合わせて変更します。

- 起動バッチファイルの変更は、TownsシステムソフトウェアのTownsMENU の「テキスト編集」でおこないます。
- 起動ドライブのルートディレクトリにある"AUTOEXEC.BAT"に、次の3つの命令を追加します。
  - ・SET FMINST （FM音源に関する設定をする命令）
  - ・SET PCMINST （PCM 音源に関する設定をする命令）
  - ・SET TOM （オンラインマニュアルに関する設定をする命令）
- 変更した内容は、FMTOWNS を起動し直さないと有効になりません。

**1 起動バッチファイルを読み込む**

①TownsMENU の［テキスト編集］を左クリックする
②メニューバー［ファイル］のサブメニューで［実行／開く］を左クリックする
③起動ドライブの "AUTOEXEC.BAT" を読み込む

- 「テキスト編集」の具体的な操作方法については、Townsシステムソフトウェアに添付のマニュアルをご覧ください。

**2 起動バッチファイルを変更する**

次の3つの命令を追加します。

・SET FMINST=("FM_1.FMB"の入っているディレクトリのパス名)
・SET PCMINST=("E_DRUMS.PMB"の入っているディレクトリのパス名)
・SET TOM=( ディレクトリ<FBM> のパス名)

例)

```
SET FMINST=I:¥BASIC  …( ﾊｰﾄﾞﾃﾞｨｽｸﾄﾞﾗｲﾌﾞ I: のBASICﾃﾞｨﾚｸﾄﾘ )
SET PCMINST=I:¥BASIC …( ﾊｰﾄﾞﾃﾞｨｽｸﾄﾞﾗｲﾌﾞ I: のBASICﾃﾞｨﾚｸﾄﾘ )
SET TOM=I:¥FBM       …( ﾊｰﾄﾞﾃﾞｨｽｸﾄﾞﾗｲﾌﾞ I: のFBMﾃﾞｨﾚｸﾄﾘ )
```

これらは "AUTOEXEC.BAT" の"CONTROL" と記述してある行の前であれば、どこに追加してもかまいません。

**3 変更した内容を保存する**

①" AUTOEXEC.BAT" を保存する
②「テキスト編集」を終了する
TownsMENU に戻ります。

**ご注意**

● この変更を失敗すると、システムが起動しなくなったり、BASIC の一部のコマンドが正常に動作しないことがあります。もし起動しない場合は、CD-ROMから起動して変更し直してください。

**ひとこと**

● "AUTOEXEC.BAT"ファイルは、"SETPATH"ユーティリティで変更することもできます。

① "SETPATH.BAS"を読み込む
F-BASIC386を起動し、[ファイル] のサブメニュー [読み込み] で、F-BASIC 386 のCD-ROMのディレクトリ<UTY> 内から"SETPATH.BAS" を読み込みます。
F-BASIC386のエディタの画面の操作については170ページをご覧ください。

②カレントディレクトリを起動ドライブのルートディレクトリに変更する
[ファイル] のサブメニュー [読み込み] で◀または▶で起動ドライブを表示します。
ルートディレクトリを表示します。
[取り消し] を左クリックします。
[SHOW ALL] で起動ドライブを表示することもできます。

③"SETPATH.BAS" を実行する

［実行］のサブメニューの［Run］を左クリックします。

"SETPATH"ユーティリティが起動します。

④Yと⏎キーを押す

⑤パス名を入力する

次の3つのメッセージが表示されます。「テキスト編集」で変更するときと同じパス名を入力してください。

・FM音源用音色ファイルのパス名？_ （I:¥BASICと入力する）

・PCM 音源用音色ファイルのパス名？_ （I:¥BASICと入力する）

・オンラインマニュアルのパス名？_ （I:¥FBMと入力する）

⑥入力したパス名を確認する

⑤で入力したパス名と、確認を求めるメッセージが表示されます。

正しく入力してあるかを確認し、Yと⏎キーを押します。

画面に表示されるメッセージに従って3つのファイルの複写先を入力すると、"AUTOEXEC.BAT"の必要な部分を変更し、"AUTO.BAT"というファイルが作られます。このファイルの名称を "AUTOEXEC.BAT" に変更して使います。ファイルの「名前変更」の具体的な操作方法については、Townsシステムソフトウェアに添付のマニュアルをご覧ください。

⑦F-BASIC386を終了する

# 4. ハードディスクからの起動と終了

## ハードディスクからの起動

F-BASIC386をハードディスクから起動するときは次のような方法で起動します。

**1 電源を入れる**

本体前面の電源スイッチを押し、電源を入れます。

F-BASIC386のCD-ROMやフロッピィディスクはセットしないでください。

しばらくするとハードディスクが起動し、TownsシステムソフトウェアのTowns MENUが表示されます。

**2 F-BASIC386のアイコンを表示する**

ドライブ選択ウィンドウで、F-BASIC386がインストールされているドライブを左クリックします。

F-BASIC386のアイコンが表示されます。

**3 F-BASIC386を選択する**

[F-BASIC386]あるいは[F-BASIC386M] のアイコンを左クリックします。

**4 F-BASIC386を実行する**

メニューバー［ファイル］のサブメニュー［実行／開く］を左クリックします。

しばらくすると、F-BASIC386のエディタ画面が表示され、F-BASIC386が使えるようになります。

## 終了の方法

ハードディスクから起動したF-BASIC386の終了の方法は、CD-ROMのときと同様です。14ページをご覧ください。

# 5. CD-ROMの内容

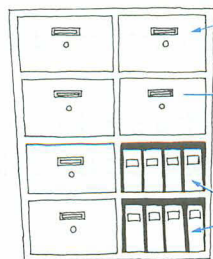
## F-BASIC386の CD-ROMの内容

F-BASIC386のCD-ROMには、F-BASIC386を動かすためのプログラムや、設定のプログラム、インフォメーションなどの他に、F-BASIC386のプログラムで利用できるように、イラストや画像、FM音源や PCM音源のサンプルデータ、ゴシック体や教科書体、丸文字体、明朝体などの書体、さらに、本マニュアルで解説しているサンプルプログラムが入っています。これらのデータやプログラムは、ディレクトリにまとめられ、ファイルの形で入っています。

# 5. CD-ROMの内容



**ディレクトリとは**

CD-ROMやフロッピィディスクをひとつの大きなキャビネットのようなものとすると、ディレクトリは引き出しにたとえることができます。引き出しには名前がついており、その中にはファイルがしまわれたり、さらに引き出し（ディレクトリ）があったりします。

＊引き出しにしまわれているプログラムやデータを取り出すには、目的のプログラムやデータが見つかるまで、次々と引き出しを開けていく必要があります。

**ファイルとは**

ファイルとは、プログラムやデータのひとつのかたまりのことで、CD-ROMやフロッピィディスク内に保存するときの最小単位です。ファイルには、ファイル名がつけられています。ファイル名は半角8文字（全角4文字）の名称と、拡張子と呼ばれるファイルの内容を表すアルファベット3文字の記号で構成されています。

ABCDEFGH.BAS
- 拡張子（BASIC のプログラムは .BAS 、絵や写真などのデータは .TIF など）
- 必ずピリオドをつける
- ファイルの名称（半角8文字、あるいは全角4文字以内）

**ひとこと**

- システムを起動した直後のTownsMENU のファイル表示ウィンドウには、アイテム登録されたファイルだけが表示されます。
- ［ドライブ選択ウィンドウ］の［表示切替えボタン］で、表示モードを「ファイル」にすると、ルートディレクトリ（前ページの図の一番左側の列）のすべての内容が表示されます。
- ファイル表示ウィンドウに表示されているディレクトリ名を左クリックで選択し、メニューバー「ファイル」のサブメニューで［実行／開く］を左クリックすると、そのディレクトリの内容が表示されます。
- 実行可能なファイル（アイテム登録されているファイルおよび.EXE,.EXP,.BAT などの拡張子がついたファイル）を左クリックで選択し、メニューバー「ファイル」のサブメニューで［実行／開く］を左クリックすると、そのプログラムが起動します。

# 6. ベーシックアイランドの冒険の使い方

**ベーシックアイランドの冒険**

F-BASIC386のCD-ROMの中には、オンラインチュートリアルマニュアル「ベーシックアイランドの冒険」が入っています。「ベーシックアイランドの冒険」では、楽しい絵や音楽を楽しみながら、F-BASIC386のプログラミングの基本が学習できます。プログラムを作る上で大切なことが説明されています。完全にマスターするように頑張りましょう。
操作は、すべて画面を見ながらできるようになっています。

**「ベーシックアイランドの冒険」を開始する**

「ベーシックアイランドの冒険」を開始するときは次のように操作します。

「ベーシックアイランドの冒険」を開始するには、2HDのフロッピィディスクが1枚必要です。

**1 選択する**　**F-BASIC386のTownsMENU の［ベーシックアイランドの冒険］を左クリックします。**

**2 実行する**　**メニューバー［ファイル］のサブメニューで［実行／開く］を左クリックします。**

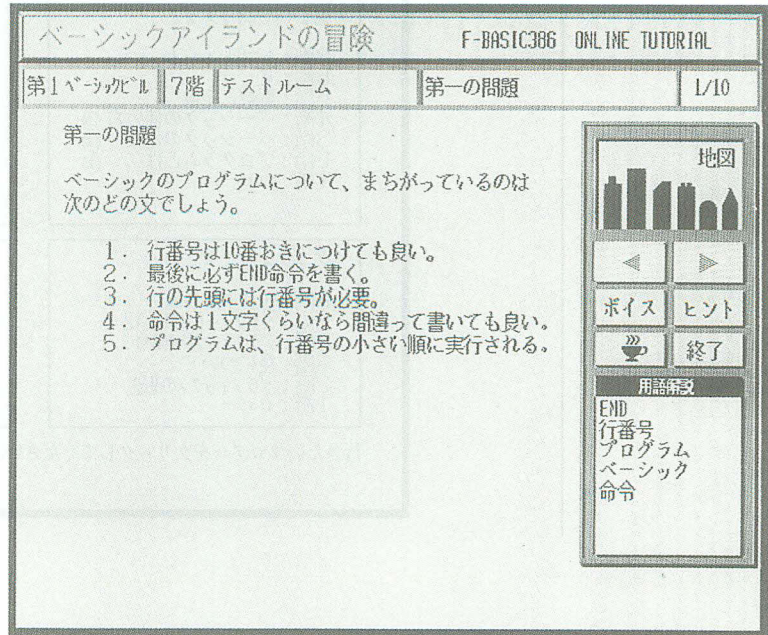
「ベーシックアイランドの冒険」の画面が表示されます。

「ベーシックアイランドの冒険」の画面

はじめて冒険をするときと、冒険をしたことがあるときで、操作の手順が違います。画面の指示に従って操作をおこなってください。

**「ベーシックアイランドの冒険」で学習する**

●冒険は「ベーシックアイランド案内図」から始まります。
●冒険は、どのビルのどのフロアーから初めてもかまいません。
●それぞれのビルには、学習の成果を試すテストルームがあります。
がんばってチャレンジしてください。

「学習のフロア」の画面

ボタンの説明

「ベーシックアイランドの冒険」にはいろいろなボタンがあります。
それぞれ次のようなはたらきをします。

［終了］　学習をやめてTownsMENUに戻ります。
［地図］　「ベーシックアイランド案内図」の画面を表示します。
［◀］　「学習のフロアー」の前ページを表示します。
［▶］　「学習のフロアー」の次ページを表示します。
［ボイス］　ナレーションを聞くことができます。
［ヒント］　学習のヒントを聞くことができます。
［☕］　学習を一時中断し、休憩の画面を表示します。
［用語解説］　用語解説の枠の中の用語をクリックすると、その用語の解説の画面を表示します。

## 6. ベーシックアイランドの冒険の使い方

### 学習の内容

●それぞれのビルは、フロアーごとに学習の内容がわかれています。

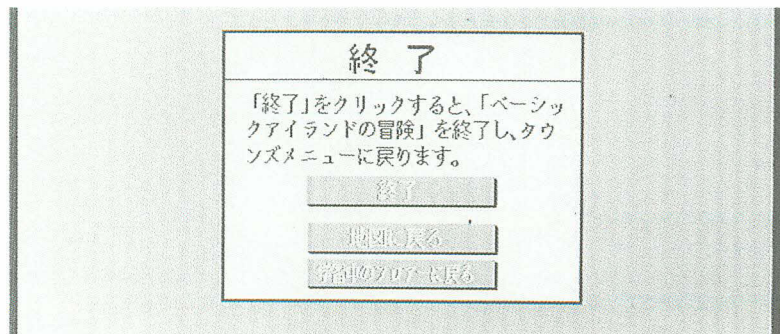
「学習カルテ」の画面

### 「ベーシックアイランドの冒険」を終了する

「ベーシックアイランドの冒険」を終了するときは次のように操作します。

**1 ［終了］を実行する**

学習のフロアの画面、または「ベーシックアイランド案内図」の画面の［終了］を左クリックします。

終了のガイドウィンドウが表示されます。

**2 終了のガイドウィンドウで［終了］を実行する**

［終了］を左クリックします。

「ベーシックアイランドの冒険」が終了し、TownsMENUに戻ります。

# 第2章

# 基本的なプログラムを作るための知識

この章では、F-BASIC386を使ってプログラムを作るために必要な、基本的な知識について説明しています。
ここで説明していることはどれも必ず知っておいていただきたいことです。
何事も最初が肝心、プログラミングの基本をしっかり勉強しましょう。

# この章をお読みになる前に

この章は、オンラインチュートリアルマニュアル「ベーシックアイランドの冒険」を終了していることを前提にしています。「ベーシックアイランドの冒険」ではプログラムとは何なのか、また、プログラムを作るためのきまりごと、そしてベーシックの命令について勉強しました。
この章では、「ベーシックアイランドの冒険」で勉強した内容に加えて、実際にプログラムを作るときに必要な、基本的なことがらについて、さらに詳しく説明します。
まず最初に、プログラムの作り方と、わかりやすいプログラムを作るための命令について説明します。
次にプログラムでおこなう基本的な処理として、数の計算、文字の処理、ファイルの処理について説明します。

## 1．プログラムの作り方

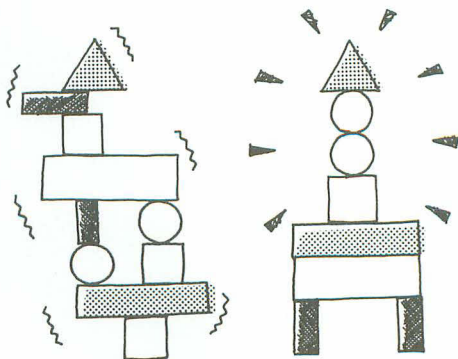
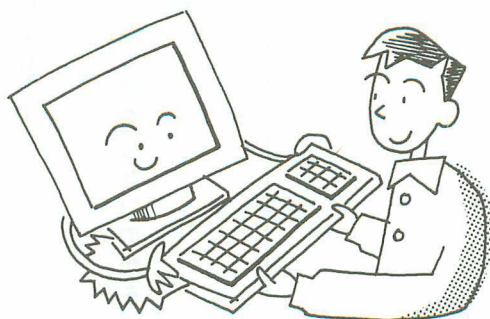
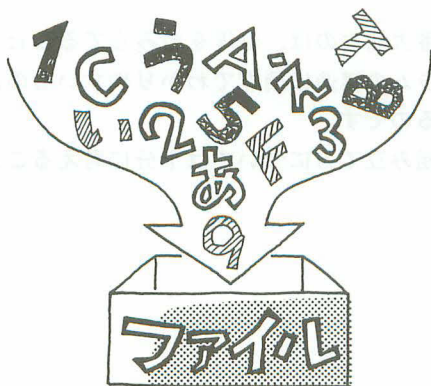
最初からいきなりF-BASIC386の命令を使いこなしてプログラムを作れる人はいません。まずはどのようにしてプログラムを作ったらいいのかを理解しましょう。

同じプログラムを作るのならわかりにくいよりも、わかりやすいプログラムを作りたいものです。わかりやすいプログラムとはすなわち無駄のない構造のプログラムです。ここでは、わかりやすいプログラムを作るために役立つ命令を紹介します。

## 2．わかりやすいプログラムを作るための命令

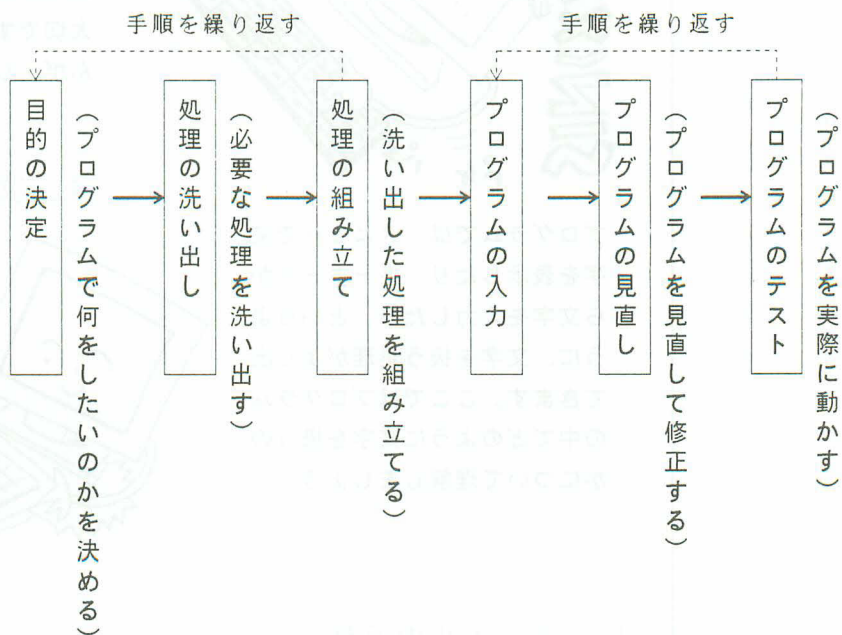
## 3．数の計算

コンピュータは、すべての処理をコンピュータ独自の方法による数の計算で実現しています。したがってプログラムを作るときは、コンピュータでおこなう数の計算をマスターすることが大切です。難しいかもしれませんが、しっかり勉強しましょう。

## 4．文字の処理

プログラムでは、メニューで文字を表示したり、キーボードから文字を入力したり、というように、文字を扱う処理がよく出てきます。ここではプログラムの中でどのように文字を扱うのかについて理解しましょう。

## 5．ファイルの処理

プログラムの中では、数や文字などのたくさんのデータを扱います。これらのデータを保存しておく便利な入れ物がファイルです。
ファイルの処理をマスターしたら、大量のデータの扱いも思いのままです。

# 1. プログラムの作り方

初めてプログラムを作るとき、「まず何から始めたらいいんだろう？」という疑問が浮かぶと思います。
プログラムは、通常次のような手順で作ります。
しかし、作っている途中で、どこかに不都合や無理が生じてしまったりして、必ずしもこの手順どおりにいかないこともあります。
そのようなときは、前の手順に戻って考え直します。

上の手順の中で、一番大切なのは、処理を組み立てることです。処理の組み立て方によって、プログラムの構造は簡潔でわかりやすいものにも、雑然としたわかりにくいものにもなるのです。
したがって、処理の組み立て方については十分に考えることが大切です。

# 1. プログラムの作り方

1

**目的の決定**

プログラムを作るときは、まず何をしたいのかという目的を決めます。この目的はできるだけ具体的に決めるようにします。目的があいまいでは、プログラムを作ることはできません。
例えば、「クラス全員のテストの平均点を求める」という目的を決めたとします。しかし、これではまだあいまいなので、次のように具体的に目的を決めます。

・クラス全員の全科目の点数を入力し、各科目ごとの平均点を求める
・クラス全員の全科目の点数を入力し、生徒各人の平均点を求める

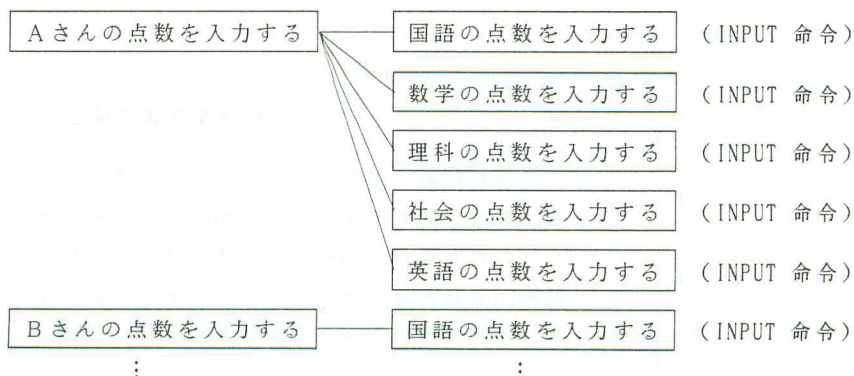
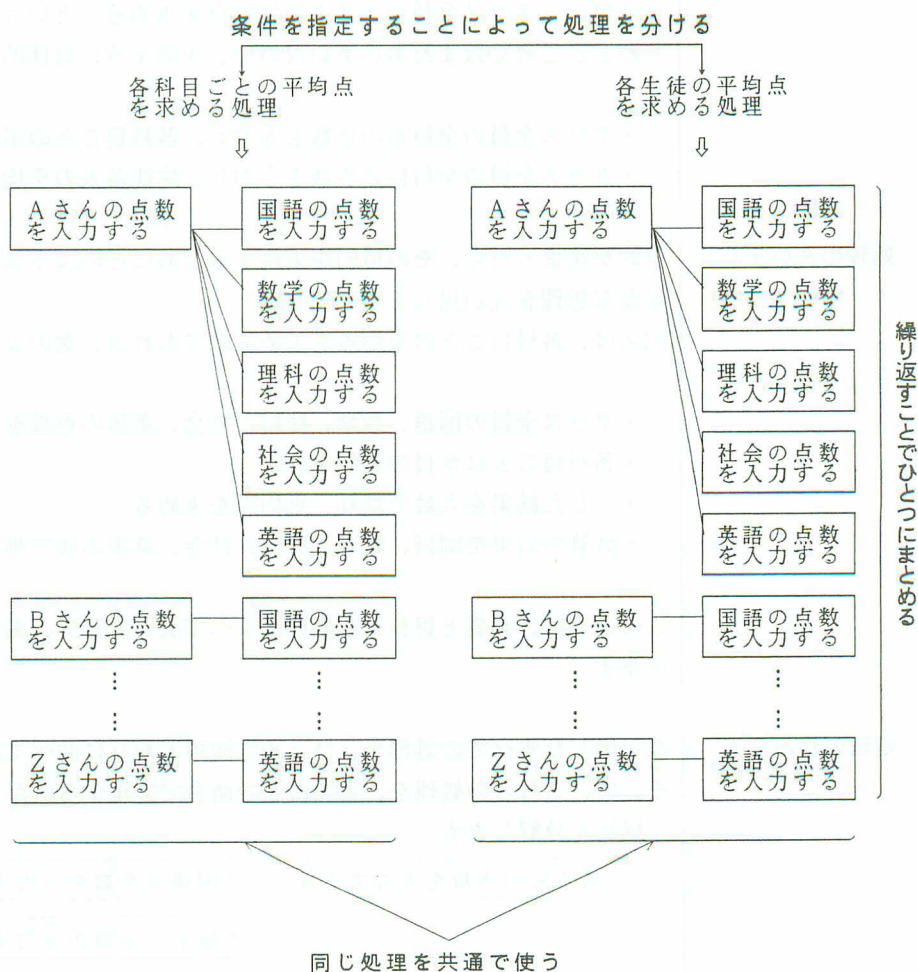
**処理の洗い出し**

目的が決まったら、その目的を実現するためにどのような処理が必要かを考え、必要な処理を洗い出します。
例えば、各科目ごとの平均点を求めるのであれば、次のような処理が必要です。

・クラス全員の国語、数学、理科、社会、英語の点数を入力する
・各科目ごとに全員の点数をたす
・たした結果を人数で割り、平均点を求める
・計算の結果を国語、数学、理科、社会、英語の順で画面に表示する

このように、必要と思われる処理はすべて洗い出して、漏れがないよう気を付けます。

**処理の組み立て**

洗い出したすべての処理を、ひとまず順番どおりに並べてみます。
そして、これらの処理を、実際にどの命令で実現するのかを考えながら、さらに細かく分解します。

そして、条件を指定することによって処理を分けたほうがいいものや、共通で使ったり、繰り返して使うことでひとつにまとめられそうなものがないかを検討しながら、処理を組み立てていきます。

そして、処理を組み立てながら、処理の流れをフローチャートに書いてみます。このとき、どうしてもうまく処理が組み立てられなかったり、流れがつながらなかったりしたときは、目的の決定や、処理の洗い出しに戻って考え直します。そして処理の流れがすべてつながって、もうこれ以上、無駄なところやまとめたりするところがない、というところまで十分に検討しつくします。

# 1. プログラムの作り方

1

| | |
|---|---|
| プログラムの入力 | 処理の組み立てが終わり、フローチャートができたら、F-BASIC386のエディタ画面で、実際にプログラムを入力します。<br>このとき、どのような変数を使うかなどを考え、リファレンスマニュアルで命令の書き方を確認しながら間違いのないように入力していきます。 |
| プログラムの見直し | プログラムの入力が終わったら、フローチャートどおりに入力できたかを確認します。もし、命令の間違いや抜けがあったら修正、追加して、フローチャートどおりのプログラムにします。 |
| プログラムのテスト | フローチャートどおりのプログラムができたら、実際にプログラムを実行してみます。<br>間違いがあったときは、エラーが発生してプログラムが止まりますので、エラーが発生した原因を探して、正しく入力し直します。そしてプログラムが完全に動くようになったら、プログラムの完成です。<br>エラーの内容や対処方法については、エラーの処理☞95ページをご覧ください。 |

# 2. わかりやすいプログラムを作るための命令

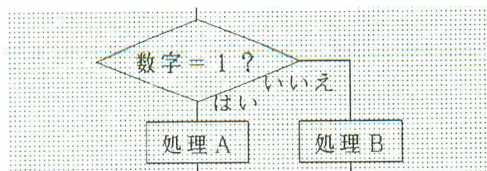
ひとつのプログラムはたくさんの処理が集まってできています。
これらの処理を組み立てるときは、次の点に注意しながら組み立てることによって、無駄を省いたわかりやすい構造のプログラムになります。

- 条件を指定することによって、処理を分けたほうがいいものがないか
- 繰り返して使うことによってひとつにまとめられる処理がないか
- 共通で使うことによってひとつにまとめられる処理がないか

これらを具体的に実現するには、次の4つの命令を使います。

- 条件分岐　　　　　　　（構造化IF文）
- 条件を指定する繰り返し（WHILE-WEND命令）
- 回数を指定する繰り返し（FOR-NEXT命令）
- サブルーチン化　　　　（GOSUB-RETURN命令）

**条件分岐（構造化IF文）**

条件分岐とは、指定した条件が成り立つか成り立たないかによって、実行する命令を変えることをいいます。
入力された文字や数字によって、実行する処理を変えたいときなどにを使います。
例えば、入力された数字が1なら、処理Aを、1以外なら処理Bを実行するというようなときは、「入力された数字が1か」という条件を指定します。そして、この条件が成り立つか、成り立たないかで、処理Aと処理Bに分かれます。
これをフローチャートで表すと、次のようになります。

条件分岐には構造化IF文またはIF-THEN-ELSE命令を使います。
どちらも同じはたらきをしますが、プログラムが見やすく、処理の流れがわかりやすい、といったことから、構造化IF文を使います。
構造化IF文とIF-THEN-ELSE命令については、☞42ページをご覧ください。

● 構造化IF文

IFの後の条件が成り立つときには、THENの後の命令が実行されます。条件が成り立たないときには、ELSEの後の命令が実行されます。

```
IF <条件> THEN
  <文 1 >  ┐
  <文 2 >  │ ブロック
  <文 3 >  ┘
ELSE
  <文 4 >  ┐ ブロック
  <文 5 >  ┘
ENDIF
```

「文」とは、命令と、その命令の対象となる変数や定数を含む、一連の表記をいいます。

「ブロック」とは、ある処理に対して関連性のある文の集まりをいいます。

IFとELSE、ELSEとENDIF の間のブロックの先頭に、それぞれ適当な数の空白を入れると、ブロックの区切りやまとまりがはっきりして、構造化IF文でおこなっている処理の流れがわかりやすくなります。これを「段下げ」といいいます。

```
IF <条件> THEN                          IF <条件> THEN
<文 1 > ┐                                 <文 1 > ┐
<文 2 > │ブロック                          <文 2 > │ブロック
<文 3 > ┘        ══════>                  <文 3 > ┘
ELSE           ブロックの先頭に            ELSE
<文 4 > ┐ブロック 空白を入れる               <文 4 > ┐ブロック
<文 5 > ┘                                 <文 5 > ┘
ENDIF                                   ENDIF
```

このガイドで紹介するプログラム例はすべて2文字の空白を入れて段下げをおこなっています。

## 2. わかりやすいプログラムを作るための命令

構造化IF文とIF-THEN-ELSE命令

構造化IF文は、F-BASIC386独自の命令で、従来のBASICでは、条件分岐にはIF-THEN-ELSE命令を使っていました。

F-BASIC386では、IF-THEN-ELSE命令も構造化IF文も使うことができます。

構造化IF文を使うと、プログラムが見やすく処理の流れがわかりやすくなります。

● IF-THEN-ELSE命令

```
IF 〈条件〉 THEN 〈文１〉:〈文２〉:〈文３〉 ELSE 〈文４〉:〈文５〉
```

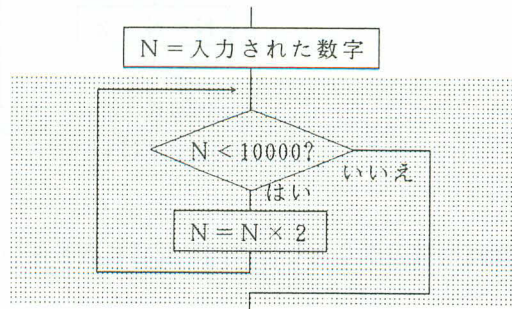
THENやELSEの後に複数の文を実行するときに、文をコロンで区切って同じ行に書かなければいけないので、文の区切りがわかりにくく、見づらい

● 構造化IF文

```
IF 〈条件〉 THEN
  〈文１〉
  〈文２〉
  〈文３〉
ELSE
  〈文４〉
  〈文５〉
ENDIF
```

THENやELSEの後に実行する文は行を変えて書くので、文の区切りが一目でわかり、見やすい

**条件を指定する繰り返し（WHILE-WEND命令）**

条件を指定する繰り返しとは、指定した条件が成り立つあいだ、処理を繰り返すことをいいます。
回数はわからないが、ある状態になるまで処理を繰り返したいときに使います。
例えば、入力された数字に2をかけて、その答えにさらに2をかけて…　という計算を、答えが10000 になるまで繰り返すようなときは、「計算の答えが10000未満か」という条件を指定します。そして、この条件が成り立つ間、計算を繰り返します。
これをフローチャートで表すと、次のようになります。

条件を指定する繰り返しには、WHILE-WEND命令を使います。

● WHILE-WEND命令

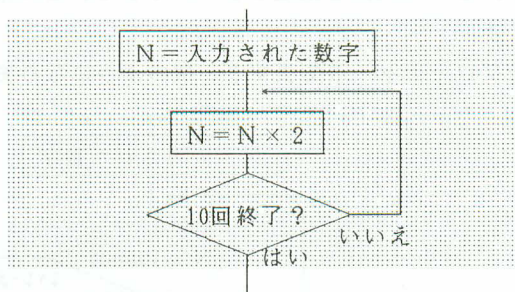
WHILE の後の条件が成り立つときには、WHILE とWENDの間のブロックが実行されます。条件が成り立たないときには、WENDの後の命令が実行されます。

```
WHILE  <条件>
  <ブロック>
WEND
```

WHILEとWENDの間のブロックは、段下げをおこなって、区切りやまとまりを見やすくします。

## 2. わかりやすいプログラムを作るための命令

**回数を指定する繰り返し**
（FOR-NEXT命令）

回数を指定する繰り返しとは、回数を決めて、処理を繰り返すことをいいます。例えば、入力された数字に2をかけて、その答えにさらに2をかけて…　という計算を10回繰り返すようなとき、回数には10回を指定します。そしてその回数だけ計算を繰り返します。
これをフローチャートで表すと、次のようになります。

指定回数の繰り返しにはFOR-NEXT命令を使います。

● FOR-NEXT命令

FOR の後に指定した回数だけ、FOR とNEXTの間のブロックが実行されます。
繰り返す回数は、「初期値」（いくつから）、「終値」（いくつまで）、「増分値」（いくつきざみで）で指定します。

```
FOR <変数> = <初期値> TO <終値> STEP <増分値>
  <ブロック>
NEXT
```

☞初期値、終値、増分値は、定数、変数、式のどれかで指定します。
☞FOR とNEXTの間のブロックは、段下げをおこなって、区切りやまとまりを見やすくします。

**WHILE-WEND命令とFOR-NEXT命令の使い分け**

WHILE-WEND命令では、条件が成り立たなければ１回もブロックは実行されませんが、FOR-NEXT命令では、変数の初期値と終値がどんな値であっても、必ず１回はブロックが実行されます。これはFOR-NEXT命令の終了判定が、NEXT命令を実行するときにおこなわれるからです。

例えば、初期値、終値、増分値を変数Ａ、Ｂ、Ｃで表し、それぞれの変数に次のような値が代入された場合を考えてみます。

変数Ａ　＝　10　…初期値
変数Ｂ　＝　５　…終値
変数Ｃ　＝　１　…増分値

これらの変数で指定される回数は、「10から５まで１きざみで」です。このような回数が指定されたときも、FOR-NEXT命令では、必ず１回はブロックが実行されてしまいます。

```
FOR　<変数> = Ａ　TO　Ｂ　STEP　Ｃ
　<ブロック>
NEXT
```

これをWHILE-WEND命令で次のように記述すると、WHILE の後の条件式が成り立たないので、ブロックは１回も実行されません。

```
WHILE Ａ　<=　Ｂ
　<ブロック>
　Ａ　＝　Ａ　＋　Ｃ
WEND
```

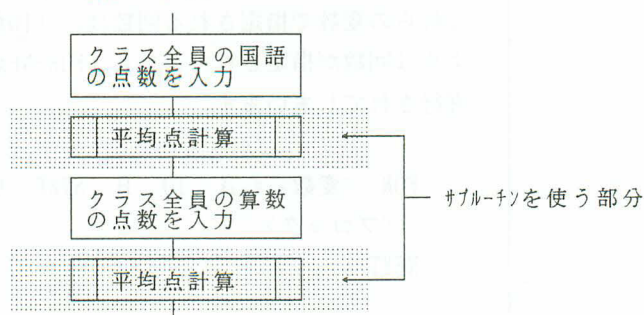
## 2. わかりやすいプログラムを作るための命令

**サブルーチン化（GOSUB-RETURN命令）**

サブルーチン化とは、プログラムの中の共通する処理を小さなプログラムにまとめることをいいます。また、まとめられたプログラムをサブルーチンといいます。サブルーチンは、プログラムのメインとなる部分（メインプログラム）とは別のところにまとめて書き、必要に応じてメインプログラムからGOSUB 命令で呼び出します。

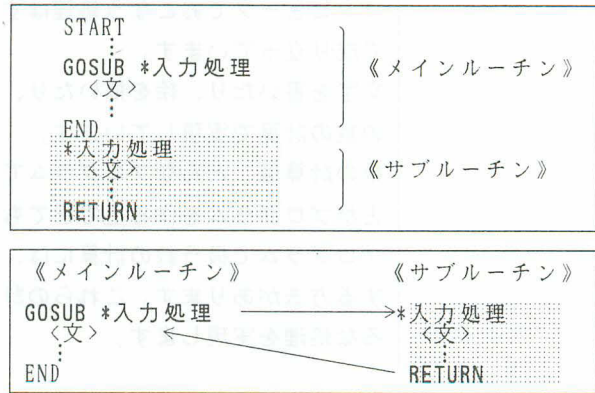
プログラムの中で、ひとつの処理がいろいろな所に共通しているようなときに使います。

例えば、クラスのテストの平均点を、科目別に求めるようなときは、平均点を求める処理（計算方法）はどの科目でも同じなので、サブルーチンにします。そして必要に応じてこのサブルーチンを呼び出します。

これをフローチャートで表すと、次のようになります。

● GOSUB-RETURN命令

メインプログラムのGOSUB命令で、サブルーチンを呼び出し、サブルーチンの先頭の文が実行されます。

サブルーチンのRETURN命令でメインプログラムに戻り、メインプログラムのGOSUB命令の次の文が実行されます。

例えば、「* 入力処理」というラベルを付けてサブルーチン化したプログラムは次のような流れで実行されます。

サブルーチンは行番号で呼ぶこともできますが、なるべく「ラベル」（名前）を付けて、そのラベルで呼ぶようにします。
ラベルを付けるときは、どのような処理をするサブルーチンなのかが一目でわかるような名前にします。
ラベルは必ず半角のアスタリスク「*」で始めて、半角文字ならアスタリスクを含めて255文字、全角文字ならアスタリスクを含めて127文字以内で指定します。
サブルーチンのラベルと、RETURNの間のブロックは、段下げをして、区切りやまとまりを見やすくします。

再帰的サブルーチン
サブルーチンの中で、さらに自分自身を呼ぶサブルーチンを、特に「再帰的サブルーチン」といいます。
例えば、ある図形を少しずつ位置を変えて10個つなげて描き、大きな図形にするようなプログラムで再帰的サブルーチンを使います。
この場合の再帰的サブルーチンは、図形を描く処理をサブルーチンにして、図形を描く位置を変えサブルーチンの中で自分自身を10回呼んで繰り返して実行するような使い方をします。
（再帰的サブルーチン☞89ページ）

# 3. 数の計算

コンピュータでおこなう処理はすべて、コンピュータ独自の方法による数の計算で成り立っています。
文字を書いたり、絵を描いたり、音楽を演奏するといった処理も、基本的にはこの数の計算で実現しています。
数の計算は、どんなプログラムでも必要な処理です。数の計算をマスターすることがプログラムを作る上でとても大切なことです。
プログラムで扱う数の計算には、算術演算、論理演算、そして関数を使って計算する方法があります。これらの計算を処理の目的に合わせて使い分けて、いろいろな処理を実現します。

## 算術演算

数の計算というと、たし算やひき算が思い浮かぶと思います。これらの数学的な計算をおこなうことを「算術演算」といいます。
算術演算の処理は7種類あります。なかでも加算、減算、乗算、除算の4つを「四則演算」といいます。それぞれの計算に用いられる、「＋」「－」「＊」「／」といった記号のことを「演算子」といいます。

| 演算子 | 意味 | 表記 | 処理 |
|---|---|---|---|
| ＋ | 加算 | A + B | A にB をたす |
| － | 減算 | A - B | A からB をひく |
| ＊ | 乗算 | A * B | A にB をかける |
| ／ | 除算 | A / B | A をB でわる |
| ^ | べき乗 | A ^ B | A をB 回かける |
| ¥ | 除算（整数） | A ¥ B | A をB でわる（整数の割り算の答えを整数で求める） |
| MOD | 剰余 | A MOD B | A をB でわった余り |

これらの演算子をいくつも組み合わせた計算式では、次のような優先順位で計算がおこなわれます。

べき乗 → 乗算、除算 → 除算（整数） → 剰余 → 加算、減算

この順位と違った計算をさせたいときには、先に計算したい式を（）でくくります。例えば、次のような式では、数字の順番で計算がおこなわれます。

**計算のスピード**

算術演算では、計算式の書き方で計算のスピードが変わりますが、F-BASIC386で計算するスピードは、他のBASIC に比べて非常に速く、簡単な計算ではスピードの差が気になることはありません。

しかし、非常に複雑な計算を大量にするようなときは、計算の書き方で生じるスピードの差は大きくなります。

次のプログラムはたし算を10万回おこなうのに要した時間を調べるプログラムです。

最初に計算開始直前の時刻を表示し、たし算を10万回おこなった後に、また時刻を表示します。これで計算をおこなうのにかかった時間がわかります。30行のたし算をいろいろな計算に変えて、他の演算のスピードを調べてみてください。計算式の書き方で処理に要する時間が違うことがわかります。

```
10 PRINT TIME$
20 FOR I=1 TO 100000
30    A = 5.0 + 5.0
40 NEXT I
50 PRINT TIME$
60 END
```

30行の置き換え例：

| | |
|---|---|
| 単なる代入文 | A = 5.0 |
| 掛け算 | A = 5.0 * 5.0 |
| べき乗（整数乗） | A = 5.0 ^5.0 |
| べき乗 | A = 5.0 ^5.1 |

☞算術演算の中で、もっとも速く計算できるのは四則演算です。

☞30行の5.0 は、実際のプログラムでは、実数型の定数であると解釈され、F-BASIC386のエディタ画面で5!と表示されます。
定数の型については、リファレンスマニュアルを参照してください。

## 3. 数の計算

論理演算

「ベーシックアイランドの冒険」で、論理式というものが出てきました。
論理式はAND 、OR、NOT などの論理演算子を使って次のように表します。

| 表記例 | 意味 |
|---|---|
| <式A> AND <式B> | 式Aが成り立ち、かつ、式Bが成り立つとき |
| <式A> OR <式B> | 式Aが成り立つか、または、式Bが成り立つとき |
| NOT <式> | 式が成り立たないとき |

これらの論理式は、構造化IF文などで条件を指定するときによく使われます。そして論理式で指定した条件が成り立つか成り立たないかは、「論理演算」という計算をおこなって判断しています。
論理演算とは、コンピュータ独自の計算方法で、ビットとビットを対応させておこなう計算です。論理式を使うときは、あまりビットのことは意識しませんが、実はそれぞれの式の値をビットという単位で考えて、計算しているのです。
ビットとは、コンピュータが計算をするときの基本となる単位で、スイッチのようなものと考えることができます。つまりコンピュータはビットというスイッチがONの状態を１で表し、OFF の状態を０で表して計算をおこないます。このように、１と０だけで表す数を２進数といいます。
論理演算には、AND、OR、NOTの演算子を使うAND演算、OR演算、NOT演算などがあります。

● AND演算

次のような４ビットの２進数ＡとＢがあったとき、AND 演算をおこなうと、結果は次のようになります。

```
        A    ……    0101
        B    ……    0110
                  ------
  A AND B    ──→   0100
```

AとBの同じ桁のビットを対応させ、両方のビットが１になっているとき（Aのビットが１、かつBのビットが１のとき）、結果は１になります。

● OR演算

ＡとＢにOR演算をおこなうと、結果は次のようになります。

```
         A    ……      0101
         B    ……      0110
                      ----
    A OR B   ──→      0111
```

AとBの同じ桁のビットを対応させ、どちらか片方のビットだけでも１になっているとき（Aのビットが１、またはBのビットが１のとき）、結果は１になります。

● NOT演算

ＡとＢにそれぞれNOT演算をおこなうと、結果は次のようになります。

```
NOT A ……  0101 ──→  1010      NOT B ……  0110 ──→  1001
               NOT                             NOT
```

ビットが逆転して、１のビットは０に、０のビットは１になります。

● 論理演算を使ったビット操作

ビットという単位で計算をおこなう、という考え方はコンピュータ独自の考え方で、プログラムの中では「ビットを操作する」という処理をよくおこないます。

例えば、第４章で紹介する「迷路ゲーム」というプログラムでは、頻繁にビットの操作をおこないます。

このプログラムでは、四角の部屋をつないで、その部屋の壁を破って道を作って迷路を描きます。

そして部屋の壁の状態を４ビットの２進数で次のように表しています。

| 左の壁が破れている状態 | 右の壁が破れている状態 | 上の壁が破れている状態 | 下の壁が破れている状態 | 左と上の壁が破れている状態 |
|---|---|---|---|---|
| =0001 | =0010 | =0100 | =1000 | =0101 |

どこかの壁を破ったときは、その壁に対応するビットを１にします。そして壁が破れているかどうかを調べるには、その壁に対応するビットが１かどうかで知ることができます。

# 3. 数の計算

● 2進数について

2進数を使った表現方法は指を折って数を数えることに例えることができます。

片手の5本の指を使って、いくつまで数を数えることができるかという質問に、通常は5までと答えるでしょう。しかし2進数を使うと、0から31まで数えることができます。

まず、5本の指で5桁のビットを表すと考えます。そして、指を伸ばした状態を0、折った状態を1とします。このように考えると、すべての指を伸ばした状態は0です。次に、親指を折った状態で1です。ここまでは普通の数え方です。しかし、2は人差し指を折って親指を伸ばします。3は人差し指と親指を折ります。4は中指を折って親指と人差指を伸ばします。このようにすると、5本の指を使って31まで表せます。

● ビットとメモリ

メモリの大きさ（容量）を表す単位に「バイト」を使います。

1バイトは8桁のビットで構成され、ビットとバイトの関係は次のようになります。

| | | |
|---|---|---|
| 8ビット | → | 1バイト |
| 1024 バイト | → | 1K（キロ）バイト |
| 1024Kバイト(1,048,576バイト) | → | 1M（メガ）バイト |

関数

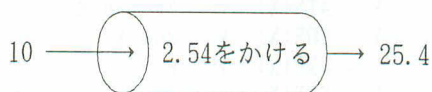
関数は複雑な計算を簡単におこなうことのできる、とても便利なものです。
F-BASIC386の関数の便利さについて説明する前に、算数で使う一般的な関数の説明をします。
関数とは、ある値に対してひとつの値が対応する関係です。すなわち、関数は、ある値を与えると、その関数で決められた計算をおこない、結果としてひとつの値を返します。関数に与える値のことを、関数の「引数」（ひきすう）といいます。そしてそれに対して返す値のことを「関数の値」または「関数の返す値」といいます。
例えば、長さの単位、インチからセンチメーターへの変換を考えてみます。1インチは2.54センチなので、10インチは2.54×10=25.4 センチになります。このときセンチは、インチを引数とする関数といえます。すなわち引数10に対して、2.54×10という関数の値が対応します。

```
2.54 ×  10  =  25.4
            └─ 引数 └─ 関数の値
```

この関数の値を求めるプログラムは次のようになります。
20行で、引数xに対応する関数の値を求めて、それを変数Yに代入しています。

```
10 INPUT "何インチですか",x
20 Y = 2.54 * x
30 PRINT "それは"; Y; " センチです。"
40 END
```

（プログラム例2-2-3 ）

この関数に、f という名前を付けます。そして、f(x)と書いて、引数x に対応する関数の値を表します。すなわち、f(x)=2.54 × xと表すことができます。

引数 : x　　関数 : f　　関数の値 : f(x)

F-BASIC386で使う関数にはユーザが自分で定義する「ユーザ定義関数」と、F-BASIC386にあらかじめ用意されている「組み込み関数」があります。組み込み関数には、数学的な計算をおこなうものや、内部の状態を知るものがあります。
ここでは、ユーザ定義関数と、組み込み関数の中でも数学的な計算をおこなう「組み込み数学関数」について説明します。

☝組み込み関数のうち、内部の状態を知るものを「システム関数」といいます。
（システム関数☞83ページ）

● ユーザ定義関数

ユーザ定義関数は、ユーザが自分で定義する関数で、プログラム中でよく使う式などを関数として定義します。ユーザ定義関数を定義するには、DEF 命令を使い、名前は必ずFNで始めます。

次のプログラム例2-2-4 は、ユーザ定義関数を使って前のプログラム例2-2-3 と同じ動作をプログラミングしたものです。 5行で、FNFという名前のユーザ定義関数を定義しています。

```
5 DEF FNF(X) = 2.54 * X
10 INPUT "何インチですか",X
20 Y = FNF(X)
30 PRINT "それは"; Y; " センチです。"
40 END
```

（プログラム例2-2-4 ）

● 組み込み数学関数

組み込み数学関数は数学的な計算をおこなうもので、算術演算を使うと計算が複雑になるものでも、この関数を使うと簡単におこなうことができます。

引数は () でくくって表記します。

組み込み数学関数には、次のような種類があります。

| 関数名 | 表記の例 | 対応する数学上の書き方 | 分類 |
|---|---|---|---|
| ABS | Y = ABS(X) | $y = \|x\|$ | 絶対値関数 |
| ATN | Y = ATN(X) | $y = \mathrm{Arctan}\ x$ | 逆三角関数 |
| COS | Y = COS(X) | $y = \cos x$ | 三角関数 |
| EXP | Y = EXP(X) | $y = e^x$ | 指数関数 |
| FIX | Y = FIX(X) | $y = (x / \|x\|) \cdot [\,\|x\|\,]$ | 整数化関数 |
| INT | Y = INT(X) | $y = [x]$ | 整数化関数 |
| LOG | Y = LOG(X) | $y = \log_e x$ | 対数関数 |
| RND | Y = RND(X) | —— | 乱数関数 |
| SGN | Y = SGN(X) | $y = x / \|x\|$ | 正負符号 |
| SIN | Y = SIN(X) | $y = \sin x$ | 三角関数 |
| SQR | Y = SQR(X) | $y = \sqrt{x}$ | 平方根 |
| TAN | Y = TAN(X) | $y = \tan x$ | 三角関数 |

（変数X を引数とし、関数の値を変数Y に代入するときの例）

チャレンジ

● 算術演算の応用例

今年の誕生日の曜日をもとに、誕生日が日曜日になるのは何年後かを求めるプログラムを作ってみましょう。

このプログラムは次のような考え方で作ります。

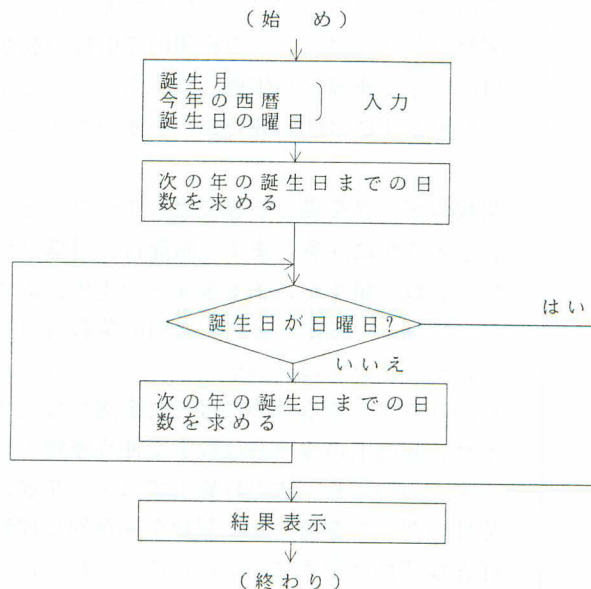
今年の誕生日から、来年の誕生日までの日数は365日です。365日を1週間の曜日の7で割った余り、これが今年と来年の誕生日の「曜日のズレ」です。割り算の余りはMOD演算で求めます。

この計算を繰り返して、誕生日がズレた結果、日曜日になるのが何年後かを求めます。

しかし、これでは、うるう年のことを考えに入れていません。うるう年の場合は1年が366日になります。

1901年から、2099年までの間のうるう年は、西暦を4で割り切れる年です。西暦を4で割ってその余りが0なら、その年はうるう年ということになります。

このように、うるう年も計算に入れて正確な日数を求めます。

こういったプログラムを作る考え方のことを「アルゴリズム」といいます。アルゴリズムとは、問題の解決法を、既知の計算を組み合わせることで実現する方法です。プログラムを作るにためには、まずその問題を解決するためのアルゴリズムを考えることが必要です。

（プログラム例2-2-5 ：誕生日が次に日曜日になる年を求める）

```
100 '-----------------------------------------------
110 ' 次の誕生日が日曜日になるのは西暦何年かを調べる
120 '-----------------------------------------------
130 INPUT " あなたの誕生日は何月ですか ",BM
140 INPUT " 今年は西暦何年ですか ",TY
150 PRINT " 今年のあなたの誕生日は何曜日ですか．数字で入力してください"
160 INPUT "日(0) 月(1) 火(2 ) 水(3) 木(4) 金(5) 土(6) ",TW
170 '
180 DD = 0
190 YY = TY
200 GOSUB * 日数を計算  'D に次の年の誕生日までの日数を求める
210 DD = DD + D: YY = YY + 1
220 WHILE (DD + TW) MOD 7 <> 0
230   GOSUB * 日数を計算 'D に次の年の誕生日までの日数を求める
240   DD = DD + D: YY = YY + 1
250 WEND
260 PRINT "次のあなたの誕生日が日曜日になるのは，西暦";YY;"年です"
270 END
280 '-----------------------------------------------
290 * 日数を計算          'D に次の年の誕生日までの日数を返す
300   IF BM <= 2 AND YY MOD 4 = 0 THEN
310     D = 366
320   ELSE IF BM > 2 AND (YY+1) MOD 4 = 0 THEN
330     D = 366
340   ELSE
350     D = 365
360   ENDIF
370 RETURN
```

● 関数のグラフ

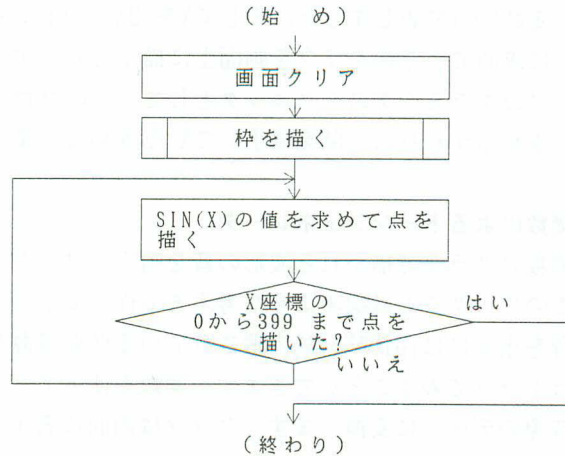
三角関数SIN(X)のグラフを描くプログラムを作ってみましょう。

このプログラムは次のような考え方で作ります。

関数のグラフは、一定の範囲内で引数を動かしながら、その引数と関数の値を対応するｘ座標とｙ座標に点を描くことによって表します。一定の範囲内で引数を動かすにはFOR-NEXT命令を使います。点を表示するにはPSET命令を使います。

関数のグラフを描くときには、グラフのどの範囲をどの程度の大きさで画面に表示するかに注意します。画面は、目盛りが１ドットの座標系と考えられ、関数の引数と値をそのまま画面上の座標にあてはめてしまうと、グラフは小さくなってしまいます。たとえばSIN 関数は最大値が１、最小値が－１ の関数なので、そのままではせいぜい２ドットか３ドットの大きさになってしまいます。わかりやすくするためにはこれを適当な大きさに拡大します。

また、画面上の座標系は数学で使う座標とは逆に、ｙ座標が下向きになっています。このため、式の計算上では＋－を逆にして座標を逆向きにします。

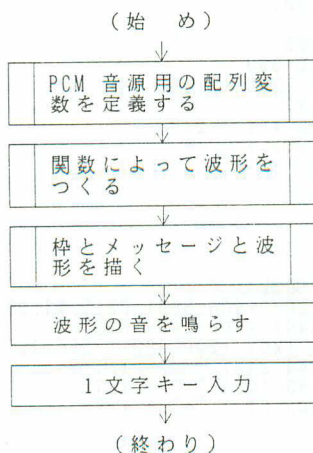
関数のグラフを描くと、関数を感覚的に理解することができます。SIN 関数を好きな関数に変えて、いろいろな関数を比べてみてください。

（プログラム例2-2-6 ：関数のグラフ）

```
100 '------------------------------------------------
110 ' 関数のグラフを描く
120 '------------------------------------------------
130 '
140 PI = 3.14159!
150 CLS
160 GOSUB *FRAME
170 FOR SX=0 TO 399
180   X = (SX - 200) / 100
190   Y = SIN(X)
200   SY = -100 * Y + 240
210   PSET(SX,SY),7
220 NEXT
230 END
240 '
250 '-----------------------------------------------
260 *FRAME                                  '枠を描く
270   COLOR 7
280   PRINT " 関数のグラフ"
290   LINE(0,120)-(400,360),PSET,%1,BF
300   LINE(0,240)-(400,240),PSET,1
310   LINE(200,120)-(200,360),PSET,1
320   SYMBOL(184,242)," O",1,1,7
330   SYMBOL(190,106),"y",1,1,7
340   SYMBOL(400,242),"x",1,1,7
350 RETURN
```

## 3. 数の計算

このプログラムでは画面上の座標を変数SXとSYで、関数の引数と関数の値を変数XとYで表しました。そしてXを-2から2まで変えながら、SIN(X)の値をYに求めて、そのグラフを画面上に描くものです。
プログラミングのテクニックとして、このプログラムではX を拡大してSXを求めるかわりに、SXを縮小してX を求めています。

● 関数によるトーンジェネレータ

関数のグラフで描かれる波形の音を鳴らすプログラムを作ってみましょう。
このプログラムは次のような考え方で作ります。
音を出すにはFM TOWNS に内蔵されているPCM 音源を使います。また、音の波形はグラフで表すことができます。関数を使ってグラフを作り、そのグラフをPCM音源のデータに変換します。グラフは画面に表示して、どんな波形なのかが目で見えるようにします。そしてPCMPLAY 命令を使って、FM TOWNS 内蔵のスピーカーから音を出します。
次のプログラム例2-2-7 は、192Hz のサイン波を出すようになっています。
サイン波はSIN 関数で作ります。SIN 関数の計算をする部分を変えると、色々な波形の音を聞くことができます。

(プログラム例2-2-7 :関数によるトーンジェネレータ)

```
100 '--------------------------------------------------
110 ' 関数によるトーンジェネレータ
120 '--------------------------------------------------
130 '    関数によって波形をつくります
140 '    波形はプログラム中で指定します
150 '    その波形の192Hz の音を出します
160 '
170 PI = 3.14159!
180 DIM WAVE%((100 + 32) / 2 - 1)
190 WPT& = VARPTR(WAVE%(0))
200 RANDOMIZE TIME
210 '
220 GOSUB *SETUP
230 GOSUB *GENERATE
240 GOSUB *DISPLAY
250 PCMPLAY WAVE%
260 A$ = INPUT$(1) : PLAY OFF
270 END
275 '-----------------------------------------------
280 *SETUP                    ' PCM音源用の配列変数の準備
290   FOR I=0 TO 7
300     POKE WPT&+I, ASC(" ")
310   NEXT I
320   WAVE%(4) = INT(RND(1)*65536) - 32768
330   WAVE%(5) = INT(RND(1)*65536) - 32768
340   WAVE%(6) = 100
350   WAVE%(7) = 0
360   WAVE%(8) = 0
370   WAVE%(9) = 0
380   WAVE%(10) = 100
390   WAVE%(11) = 0
400   WAVE%(12) = &H4B00
410   WAVE%(13) = 0
420   WAVE%(14) = 48
430   WAVE%(15) = 0
440 RETURN
450 '-----------------------------------------------
455 *DISPLAY                              ' 枠と波形の表示
460   CLS
470   PRINT " 関数によるトーンジェネレータ"
480   PRINT " 何かキーを押すと止まります"
490   LINE(268,110)-(372,370),PSET,%1,BF
500   LINE(268,110)-(372,370),PSET,1,B
510   FOR I=0 TO 99
520     W = PEEK(WPT& + 32 + I)
530     IF W >= &H80 THEN W = - (W AND &H7F)
540     LINE(270+I,240)-(270+I,240-W),PSET,7
550   NEXT I
560 RETURN
570 '-----------------------------------------------
575 *GENERATE                   ' 関数によって波形をつくる
580   FOR I=0 TO 99
590     T = I / 100
600   ' W = 0                                          ' 無音
610     W = 127 * SIN(2 * PI * T)                    ' 正弦波
620   ' M = 2!: W = 127 * SIN(2*PI*(T+M*SIN(2*PI*T))) 'FM 変調
630   ' R = .5: W = 127 * (2 * (T < R) + 1)           ' 矩形波
640   ' W = 2 * 254 * ABS(T - .5!) - 127             ' 三角波
650   ' W = 254 * (T - .5!)                          ' のこぎり波
660   ' W = 254 * RND(1) - 127                       ' ノイズ
670     IF W < 0 THEN W = &H80 OR (-W)
680     POKE WPT& + 32 + I, W
690   NEXT I
700 RETURN
710 '
```

# 4. 文字の処理

コンピュータの世界ではアルファベットや漢字、ひらがななどのすべての文字は数で表され、ひとつひとつの文字に、それぞれ異なる数値が割り当てられています。例えばアルファベットの"A"は65という数値で、数字の"1"は49という数値で表されます。しかし、プログラムを作るときに、文字をすべて数値にして処理するのは大変です。そのようなとき、文字列変数、文字列定数、文字を扱う関数を使うと、文字を数値にせずに処理することができます。

## 文字の表しかた

文字は1バイト（半角）の数値または2バイト（全角）の数値で表され、その1バイトまたは2バイトの数値を「コード」といいます。
文字を扱う命令や関数では、この、バイトとコードを指定する必要があります。
コードには次のような種類があります。

● ASCII （アスキー）コード

ANSI（米国規格協会）で定められたコンピュータ用の文字コードです。アルファベット、記号、数字に1バイトの数値を割り当てたもので、日本語は扱えません。

● JIS コード

JIS （日本工業規格）で定められたコンピュータ用の文字コードです。ASCIIコードを改良して日本語も表せるようにしたものです。
JIS コードで表す文字は、「ANK文字」と「日本語文字」に分けられます。
ANK 文字は1バイトの数値で表され、半角のアルファベット、記号、数字、カタカナが含まれます。
日本語文字は2バイトの数値で表され、全角のアルファベット、記号、数字、カタカナ、ひらがな、漢字、ギリシャ文字、ロシア文字が含まれます。
ANK 文字と日本語文字が混在する場合は、日本語文字の始まりと終わりを特別な記号で区切る必要があります。

● シフトJIS コード

ANK 文字と日本語文字が混在できるように、JIS コードを改良したコードです。シフトJIS コードでは、日本語文字の先頭の1バイトがANK 文字のコードでは未定義の81以上の数値になっており、区切り記号がなくてもANK 文字と日本語文字との区別がつくようになっています。

ASCIIコードで表される文字とANK 文字をキャラクタコードといいます。FMTOWNSでは、キャラクタコードとシフトJIS コードを使います。

# 4. 文字の処理

**文字列定数／文字列変数**

F-BASIC386では、文字を扱う変数や定数を、数値と区別して、文字列変数、文字列定数といいます。

文字列とは、文字がいくつかつながったものをいいます。（文字が1つの場合も文字列といいます）

- 「文字列定数」 — 特定の文字列を表記するためのもので、その文字列をダブルクォーテーションマーク「"」でくくって表記します。
  数字も「"」でくくると、文字とみなされます。
  なにもくくらずに、「""」と表記すると、文字が0個の文字列になります。これを「ヌル文字列」といいます。
- 「文字列変数」 — 文字列を保持するための変数で、変数名の最後にダラーマーク「$」を付けて表記します。

☞ヌルはドイツ語でゼロの意味です。ヌル文字列は、文字列を初期化する（ゼロの状態にする）ときなどに使います。

☞文字列は＋の演算子を使ってつなぎ合わせることができます。これを文字列の連接といいます。

☞文字に割り当てられている数値を、小さい順に並べると、すべての文字はコード順に並びます。したがって文字列の大小の比較は、数値の大小の比較と同じようにおこなうことができます。
比較をおこなうには、数値の比較と同様に、関係演算子「<」「>」「<＝」「>＝」「＝」「<>」を使います。

## 4. 文字の処理

**文字を扱う関数**

文字を扱う関数には、次のようなものがあります。

● LEFT$ (<ANK文字列>, <ANK文字数>)
　ANK文字列の左から、指定された数の文字列を返す。

● KLEFT$ (<日本語文字列>, <日本語文字数>)
　日本語文字列の左から、指定された数の文字列を返す。

● MID$ (<ANK文字列>, <位置>, <ANK文字数>)
　ANK文字列の指定された位置から指定された数の文字列を返す。

● KMID$ (<日本語文字列>, <位置>, <日本語文字数>)
　日本語文字列の指定された位置から指定された数の文字列を返す。

● RIGHT$ (<ANK文字列>, <ANK文字数>)
　ANK文字列の右から、指定された数の文字列を返す。

● KRIGHT$ (<日本語文字列>, <日本語文字数>)
　日本語文字列の右から、指定された数の文字列を返す。

● LEN (<ANK文字列>)
　ANK文字列の文字数を返す。

● KLEN (<日本語文字列>)
　日本語文字列の文字数を返す。

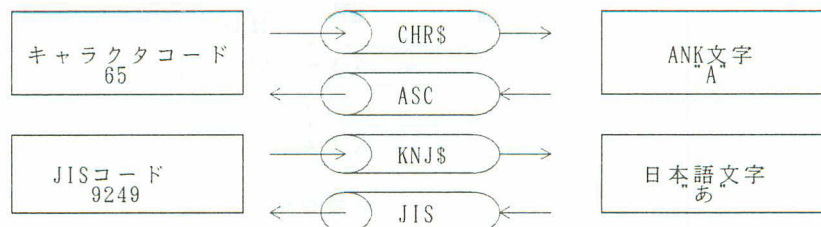
● INPUT$ (<ANK文字数>)
キーボードから入力されたANK文字を返す。

☞LEFT$関数、MID$関数、RIGHT$関数、LEN関数では、日本語文字は2バイトのANK文字と判断されます。
KLEFT$関数、KMID$関数、KRIGHT$関数、KLEN関数では、ANK文字も日本語文字も1文字を1と判断されます。

☞INPUT$関数では日本語文字は2バイトのANK文字と判断されます。

転換関数
一般に、文字列変数や文字列定数、文字を扱う関数を使った処理では、バイト数やコードを意識することはあっても、数値そのものを意識することはありません。しかし、文字と数値の対応を考える必要がある場合もあります。
例えば、ひらがなからカタカナへ変換するようなプログラムでは、次のような処理をおこないます。
カタカナのJIS コードは、ひらがなのJIS コードにちょうど&H100（F-BASIC386では「&H」は16進数であることを表します）をたしたものなので、文字（ひらがな）をいったん数値に転換して&H100 をたした後、数値から文字（カタカナ）に転換します。
このように、文字と数値を転換するには、転換関数という関数を使います。
転換関数にはキャラクタコードとJISコードで、それぞれ次のようなものがあります。

＜キャラクタコード＞
- CHR$(<数値または数値変数>) — 引数に与えたキャラクタコードの数値に対応する文字を返す
- ASC（<文字または文字変数>）— 引数に与えた文字に対応するキャラクタコードの数値を返す（引数に２文字以上の文字列を与えた場合、常に最初の１文字が転換の対象になる）

＜JIS コード＞
- KNJ$(<数値または数値変数>) — 引数に与えたJISコードの数値に対応する文字を返す
- JIS（<文字または文字変数>）— 引数に与えた文字に対応するJISコードの数値を返す

これらの関数の引数に、文字列を指定しても、最初の１文字しか転換されません。

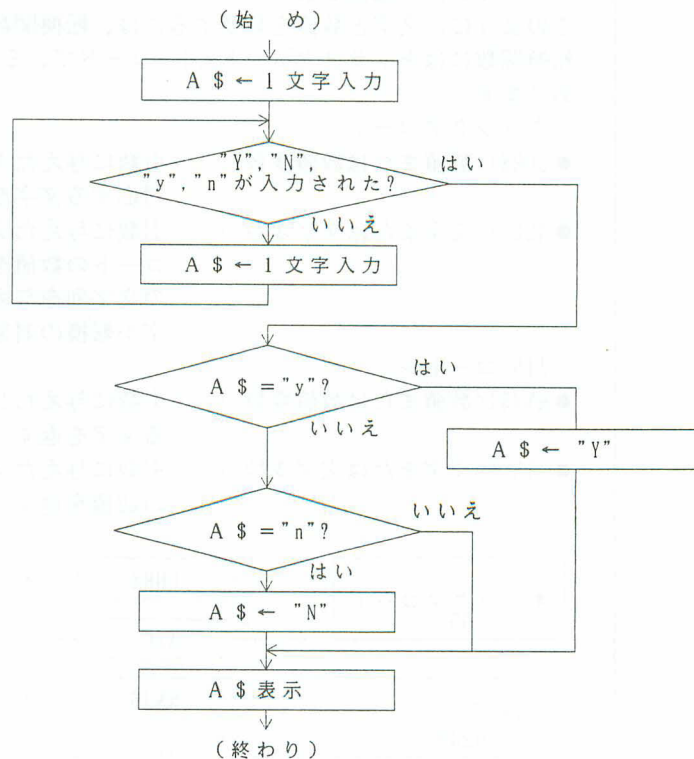
## 4. 文字の処理

チャレンジ

● YES/NOの選択

YESかNOをキーボードから入力して、入力された文字を表示するプログラムを作ってみましょう。

このプログラムは次のような考え方で作ります。

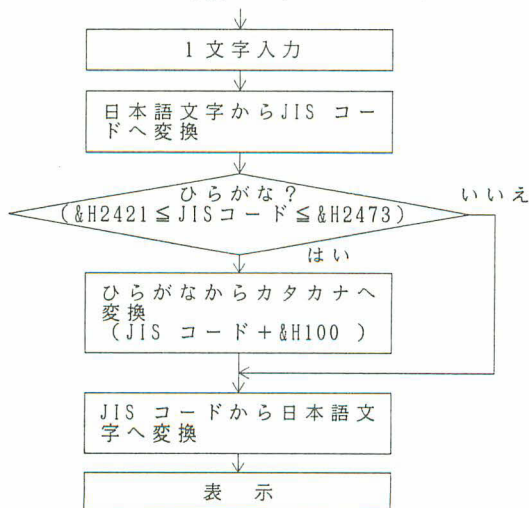
キーボードからYまたはNが入力されたら、文字列変数A$に文字列"Y"か"N"を代入します。また、小文字のyまたはnが入力されたときには、それを大文字に変換して文字列変数A$に代入します。それ以外の文字が入力されたときはWHILE-WEND命令を使って、ループを繰り返し、YNynのどれかが入力されるまで待ちます。

（プログラム例2-3-1 ：YES/NOを選択するプログラム）

```
100 '---------------------------------------------
110 ' YES/NOを選択するプログラム
120 '---------------------------------------------
130 PRINT " よろしいですか(Y/N) ";
140 GOSUB *YESNO
150 END
160 '
170 '---------------------------------------------
180 *YESNO                          ' Y/N入力ルーチン
190   A$ = INPUT$(1)
200   WHILE NOT( A$ = "Y" OR A$ = "y" OR A$ = "N" OR A$ = "n" )
210     A$ = INPUT$(1)
220   WEND
230   IF A$ = "y" THEN A$ = "Y"
240   IF A$ = "n" THEN A$ = "N"
250   PRINT A$
260 RETURN
```

### ●ひらがなからカタカナへの変換

ひらがなからカタカナへ１文字変換するプログラムを作ってみましょう。

このプログラムは次のような考え方で作ります。

ひらがなに割り当てられているJIS コードは、&H2421から、&H2473までです。そしてカタカナに割り当てられているJIS コードは、ひらがなのJIS コードにちょうど&H100 を足したものです。変換にはこの規則を使います。まず、JIS関数を使って日本語文字をJIS コードの数値に変換します。そしてその文字がひらがなであれば、その数値に&H100 をたしてカタカナの数値に変換します。最後にKNJ$関数を使って、JIS コードの数値を日本語文字に戻します。

（プログラム例2-3-2 ：ひらがなからカタカナへ変換するプログラム）

```
100 '--------------------------------------------------
110 ' ひらがなからカタカナへ変換するプログラム
120 '--------------------------------------------------
130 INPUT " ひらがなを入力してください = " A$
140 CODE = JIS(A$)
150 IF CODE >= &H2421 AND CODE <= &H2473 THEN CODE = CODE + &H100
160 A$ = KNJ$(CODE)
170 PRINT A$
180 END
```

# 5. ファイルの処理

プログラムでは、数字や文字などのたくさんのデータを扱いますが、これらのデータは電源を切ったりプログラムを終了すると消えてしまいます。これらを消さずにとっておくには、フロッピィディスクやハードディスクにファイルを作り、その中に保存します。
データをファイルに保存したり、取り出したりするには、次のような処理をおこないます。

●ファイルにデータを保存するとき
　ファイルをオープンする
　ファイルにデータを書き込む
　ファイルをクローズする
●ファイルからデータを取り出すとき
　ファイルをオープンする
　ファイル内のデータを読み込む
　ファイルをクローズする

ファイルについては、☞27ページをご覧ください。
ファイルにデータを書いたり読んだりすることを「ファイルの入出力」または「ファイルのアクセス」といいます。
ファイルにはデータ以外にプログラムも保存できます。
プログラムをファイルに保存したり、取り出したりするには、F-BASIC386のエディタ画面で、[保存]や[読み込み]の操作をおこないます。
（エディタの操作☞ 185ページ）

## 5. ファイルの処理

**ファイルのオープン**

新たにファイルを作るときや、すでにあるファイルのデータを取り出すときは、まずファイルをオープンします。

ファイルをオープンするときはOPEN命令を使います。

・OPEN命令 —— これから処理をおこなうファイルにファイル番号を割り当てます。

OPEN <モード>,<ファイル番号>,<ファイル名>

| OPEN "O",#1,"KEYFILE" | …KEYFILE という名前のファイルを新しくオープンし、ファイル番号１を割り当てる |
|---|---|

モードはファイルの用途を指定するものです。新しいファイルを作るのか、すでにあるファイルをオープンするのかなどの指定をおこないます。
モードには、"I"、"O"、"A"、"R"があります。詳しくはF-BASIC386リファレンスガイドを参照してください。

ファイル番号はファイルのアクセスのために用いる番号で、ファイルをどこまで書いたか、どこまでを読んだかといったことを管理するために必要なものです。ファイル番号は、ファイルごとに違った番号を割り当てます。
オープンした後のファイルのアクセスは、ファイル名でなく、ファイル番号を指定しておこないます。
ファイル番号は１～16の範囲で指定します。

**ファイルのクローズ**

ファイルをオープンしたら、必ずクローズします。ファイルをオープンしたままフロッピィディスクを交換したり電源を切ったりすると、ファイル内のデータが壊れてしまうことがあります。

ファイルをクローズするときはCLOSE命令を使います。

・CLOSE命令 —— データの書き込みが終わっていない場合は書き込みを終え、OPEN命令で割り当てたファイル番号を開放し、別のファイルのアクセスのためにその番号を使えるようにします。

CLOSE <ファイル番号>

| CLOSE #1 | …ファイル番号１のファイルをクローズする |
|---|---|

**ファイルへのデータの書き込み**

ファイルにデータを保存するときは、どのファイルにどのようなデータを保存するのかを指定して書き込みます。

ファイルへデータを書き込むときは、書き込むデータが文字列でも、数値でも、PRINT#命令を使います。

ファイルには「順ファイル」と「ランダムファイル」の2種類がありますが、ここでは、順ファイルへ書き込む場合を説明します。

● 文字列を書き込む場合

・PRINT#命令 —— 文字列変数や文字式に代入されている文字列を書き込みます。文字列を区切るため、文字列変数や文字式の前後をダブルクォーテーションマーク(CHR$(34))でくくり、間を＋の演算子でつなぎます。

PRINT# <ファイル番号>,"<文字列変数>"

| | |
|---|---|
| PRINT #1,CHR$(34)+A$+CHR$(34) | …文字列変数 A$ の値をファイル番号1のファイルに書く |

1行の文字列として書き込むときは、セミコロンを付けます。そうすると文字列の最後に、行の区切りであることを表す「CR」(キャリッジリターン)と「LF」(ラインフィード)という文字を付けて書き込みます。

```
PRINT #1,CHR$(34)+A$+CHR$(34) ;
                              └─セミコロン
```

● 数値を書き込む場合

・PRINT#命令 —— 数値変数や数値式に代入されている数値を書き込みます。文字列を書き込むときに使ったダブルクォーテーションマークは付けません。

PRINT# <ファイル番号>,<数値変数>

| | |
|---|---|
| PRINT #1,A | …数値変数 A の値をファイル番号1のファイルに書く |

☞順ファイルとランダムファイルについては、☞71ページをご覧ください。
☞「CR」、「LF」については☞71ページをご覧ください。

# 5. ファイルの処理

**ファイル内のデータの読み込み**

ファイルに保存されているデータを取り出すには、どのファイルからどのくらいの長さの文字列を取り出すのかを指定して変数に読み込みます。

ファイルからデータを読み込むときは、読み込むデータが文字列のときは、INPUT$関数、INPUT#命令、LINE INPUT# 命令を使います。読み込むデータが数値のときは、INPUT#命令を使います。

ファイルには「順ファイル」と「ランダムファイル」の2種類がありますが、ここでは、順ファイルのデータを読み込む場合を説明します。

●文字列を読み込む場合

・INPUT$関数 —— 指定した文字数だけ読み込みます。

INPUT$ (<文字数>,<ファイル番号>)

```
A$ = INPUT$(1,2)    …ファイル番号2のファイルから一文字を文字列変
                      数A$に読む
```

・INPUT#命令 —— 「CR/LF」, 「:」, 「,」の記号の前までの文字列を読み込みます。

INPUT# <ファイル番号>,<文字列変数>

```
INPUT #1, A$        …ファイル番号1のファイルから文字列を文字列変
                      数A$に読む
```

・LINE INPUT# 命令 —— 1行の文字列（「CR/LF」の前までの文字列）を読み込みます。

LINE INPUT# <ファイル番号>,<文字列変数>

```
LINE INPUT #1,A$    …ファイル番号1のファイルから一行を文字列変数
                      A$に読む
```

●数値を読み込む場合

・INPUT#命令―「CR/LF」,「:」,「.」の記号の前までの数値を読み込みます。
INPUT# <ファイル番号>,<数値変数>

| INPUT #1, A …ファイル番号1のファイルから数値を数値変数A に読む |
|---|

ファイルの最後まで読み終わったかどうかを調べるにはEOF関数を使います。EOF関数はファイルの終わりを検出する関数で、ファイルの最後まで読み終わったときは、－1を返し、読み終わっていないときは0を返します。

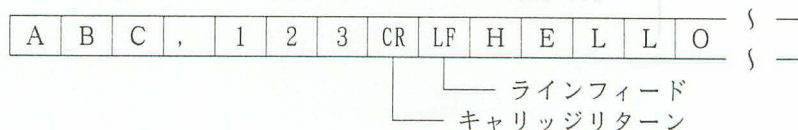
**ファイル内のデータ**

ファイルには、「順ファイル」と「ランダムファイル」の2種類があり、ファイルを作るときに、どちらのファイルにするかを指定します。

順ファイルは、先頭から順にデータを書き込み、読み込むときも先頭から読み込みます。ランダムファイルはデータをある一定の長さで管理し、先頭から何番目かを指定して書き込み、読み込みをおこないます。ランダムファイルは、データのアクセスをするときに、そのつどデータのある位置を探す時間がかかるので、通常は順ファイルを使います。

順ファイルの中のデータは、保存した順に、すきまなく、びっしりと詰まっています。ファイル内のデータは保存した順に並んでいるので、取り出すときは保存した順番と一致するようにしなければいけません。ファイルの先頭から順に保存したデータは、先頭から取り出します。真ん中から先頭、そして最後を取り出す、などということはできません。

行の区別は「キャリッジリターン」と「ラインフィード」という、2つの特別な文字によっておこないます。キャリッジリターンは略して「CR」と表し、キャラクタコードは13です。ラインフィードは略して「LF」と表し、キャラクタコードは10です。

ファイル内のデータは次のように保存されています。

# 5. ファイルの処理

チャレンジ

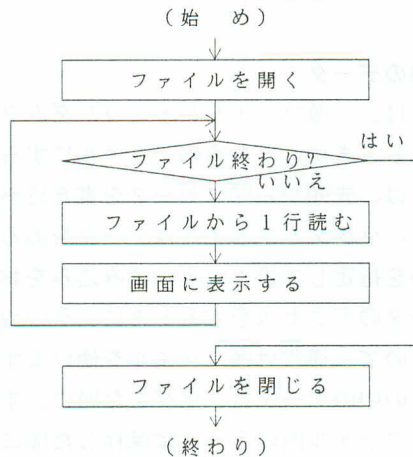
● ファイル内容を表示するプログラム

ファイルの内容を読み込んで表示するプログラムを作ってみましょう。

このプログラムは次のような考え方で作ります。

ファイルをオープンし、一行を読んで表示するという作業をファイルの終わりまで繰り返します。そして最後にファイルをクローズします。

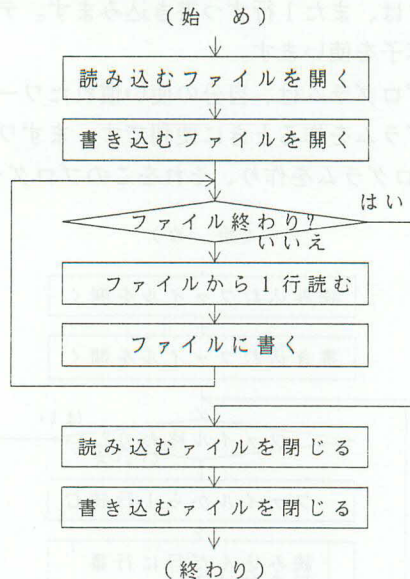
ファイルの終わりを調べるのにはEOF関数を使います。

このプログラムを使えば、文字列や数値が実際にファイルの中にどんなふうに書かれているのかを確かめることができます。

（プログラム例2-4-1 ：ファイル内容表示）

```
100 '-----------------------------------------------
110 ' ファイル内容を画面に表示する
120 '-----------------------------------------------
130 '
140 INPUT "表示するファイルは？",FLNAME$
150 OPEN "I", #1, FLNAME$
160 WHILE NOT EOF(1)
170   LINE INPUT #1, ONELINE$
180   PRINT ONELINE$
190 WEND
200 CLOSE #1
210 END
```

●ファイルをコピーするプログラム

ファイルをコピーして同じファイルを2つ作るプログラムを作ってみましょう。

このプログラムは、前のプログラム例2-4-1 に、次のような処理を加えます。書き込み用に新しくファイルをオープンし、そのファイルに読み込んだデータを一行ずつ書き込みます。そして最後にファイルをクローズします。

(プログラム例2-4-2 :ファイルをコピーする)

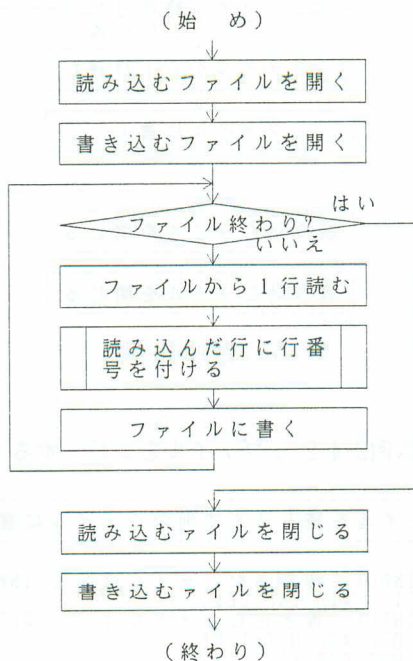
```
100 '----------------------------------------------
110 ' ファイルを読み込んで別のファイルに書き込む
120 '----------------------------------------------
130 '
140 LINE INPUT "読み込むファイルは?", INFILE$
150 OPEN "I", #1, INFILE$
160 LINE INPUT "書き込むファイルは?", OUTFILE$
170 OPEN "O", #2, OUTFILE$
180 WHILE NOT EOF(1)
190   LINE INPUT #1,L$
200   PRINT #2, L$
210 WEND
220 CLOSE #1
230 CLOSE #2
240 END
```

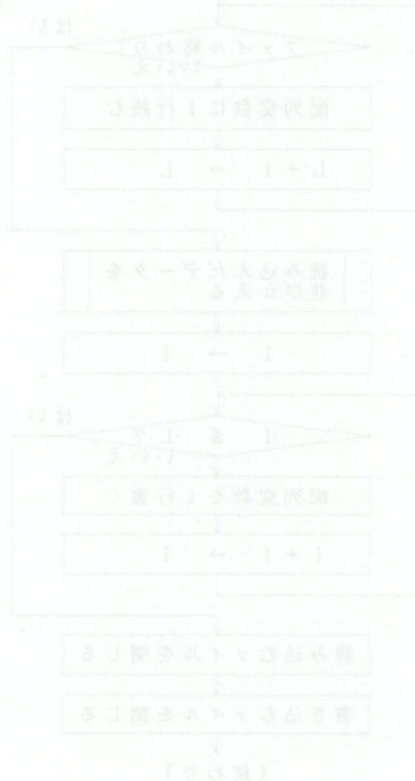
## 5. ファイルの処理

●ファイルのデータに行番号を付けるプログラム

データの行の先頭に、BASICのプログラムで使うような行番号を付けるプログラムを作ってみましょう。

このプログラムは、前のプログラム例2-4-2 に、次のような処理を加えます。データを1行読むごとに、行番号を付ける処理をおこないます。行番号を付けたデータは、また1行ずつ書き込みます。データに行番号を付けるためには、＋の演算子を使います。

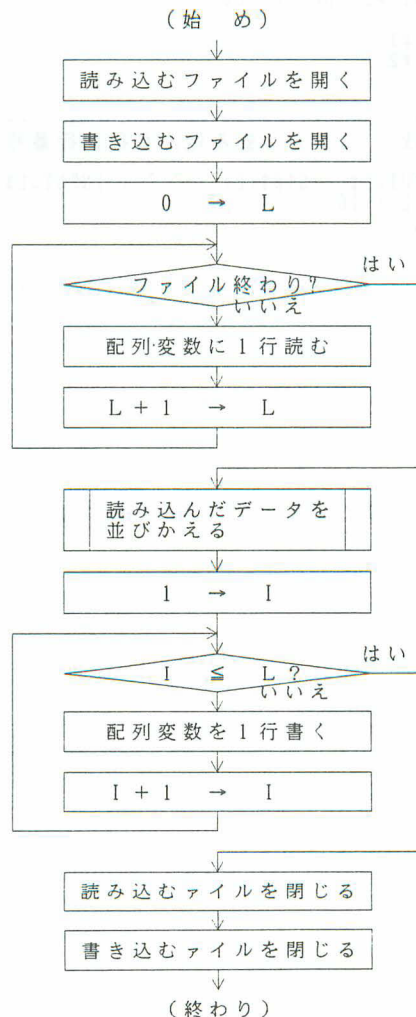
このプログラムは、自分の使い慣れたワードプロセッサなどを使ってBASICプログラムを作るときに便利です。まずワードプロセッサで行番号のないBASICプログラムを作り、それをこのプログラムを使って行番号を付けます。

（プログラム例2-4-3 ：行番号をふる）

```
100 '--------------------------------------------
110 ' ファイルに行番号を付けて別のファイルに書き込む
120 '--------------------------------------------
130 '
140 LINE INPUT "行番号を付けるファイルは？", INFILE$
150 OPEN "I", #1, INFILE$
160 LINE INPUT "書き込むファイルは？", OUTFILE$
170 OPEN "O", #2, OUTFILE$
180 L = 100
190 WHILE NOT EOF(1)
200   LINE INPUT #1, INPUT-L$
210   GOSUB *NUMBER
220   PRINT OUTPUT-L$
230   PRINT #2, OUTPUT-L$
240 WEND
250 CLOSE #1
260 CLOSE #2
270 END
280 '
290 '--------------------------------------------
300 *NUMBER     '読み込んだ行に行番号を付ける
310 '
320   OUTPUT-L$ = STR$(L) + " " + INPUT-L$
330   L = L + 10
340 RETURN
```

## 5. ファイルの処理

●ファイルの行を並びかえるプログラム

ファイル内のデータを行単位でコード順に並びかえるプログラムを作ってみましょう。

このプログラムは次のような考え方で作ります。

データをコード順に並べるには、いったんファイル内のデータを全部読み込まなければいけません。そのため、あらかじめ文字列の配列変数に、すべての行を読み込んでから処理をおこないます。処理がすべて終わったら、配列変数をファイルに書き込みます。

L＝ファイルから配列変数に読み込んだ行数を数える変数

I＝配列変数からファイルに書き込んだ行数を数える変数

（プログラム例2-4-5 : 行単位の並びかえ）

```
100 '---------------------------------------------------
110 ' ファイルを行単位でソートして別のファイルに書き込む
120 '---------------------------------------------------
130 '
140   DIM TLINE$(1000)
150   '
160   LINE INPUT "ソートするファイルは？",INFILE$
170   OPEN "I", #1, INFILE$
180   LINE INPUT "書き込むファイルは？",OUTFILE$
190   OPEN "O", #2, OUTFILE$
200   '
210   L = 0
220   WHILE NOT EOF(1)
230     LINE INPUT #1, TLINE$(L+1)
240     L = L + 1
250   WEND
260   '
270   GOSUB *SORT
280   '
290   I = 1
300   WHILE I <= L
310     PRINT TLINE$(I)
320     PRINT #2, TLINE$(I)
330     I = I + 1
340   WEND
350 END
360 '
370 '---------------------------------------------------
380 *SORT            ' バブルソート法により並びかえをおこなう
390   I = 0
400   WHILE I < NLINES
410     J = I
420     WHILE J < NLINES
430       IF TLINE$(I) > TLINE$(J) THEN SWAP TLINE$(I), TLINE$(J)
440       J = J + 1
450     WEND
460     I = I + 1
470   WEND
480 RETURN
```

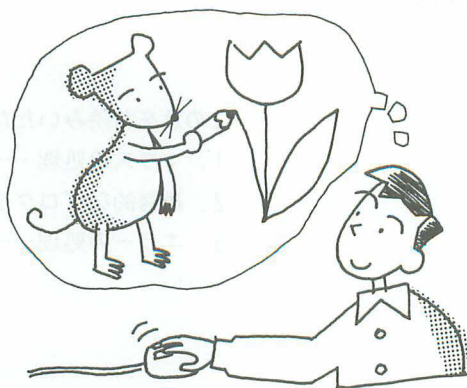
# 第3章

## 本格的なプログラムを作るための知識

この章では、より本格的なプログラムを作るための知識について説明しています。
難しいかもしれませんが、少しでもわかってきたらしめたものです。きっと、プログラムを作る素晴らしさを味わっていただけると思います。

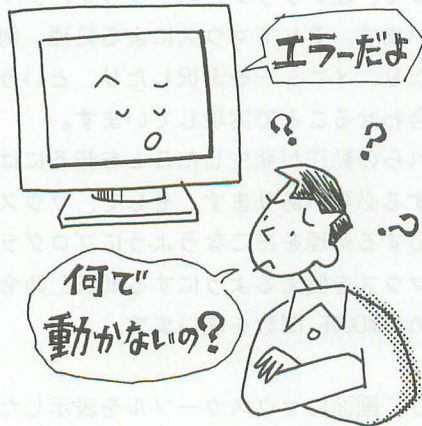
# この章をお読みになる前に

第2章では、プログラムを作るために必要な、基本的なことがらについて説明しました。それらのことを理解できていれば、基本的なプログラムを作ることはできます。しかし、次の第4章で紹介するような本格的なプログラムを作るには、第2章で勉強した内容に加えて、もう少し勉強が必要です。
この章では、マウスを扱う処理と、知っていると必ず役に立つプログラミングのテクニック、再帰的なプログラムと、エラーの処理について説明します。

## 1．マウスの処理

マウスが使われだしたのは比較的最近のことですが、ゲームや絵を描くプログラムでは、今やマウスは必需品です。ここでは、どのようにマウスの動作を検出し、その結果でどのようなプログラムができるのかを説明します。

## 2．再帰的なプログラム

本格的なプログラムを作るために、少し高度なテクニック、再帰的なプログラムを紹介します。このテクニックは、次の第4章で紹介するサンプルプログラムでも随所に使われています。

## 3. エラーの処理

プログラムに何らかの間違いがあるとエラーが発生して、プログラムが止まってしまいます。しかしエラーが発生したからといってコンピュータ本体やプログラムが壊れてしまうわけではありません。エラーが発生したら、その原因を修正してエラーが発生しないようにすればいいのです。ここでは、いろいろなエラーに対する処理について説明します。

# 1. マウスの処理

マウスは、マウスの中にある球で、画面上の動きを検出し、2つのボタンのどちらかを押すことによって、左クリック、右クリック、ダブルクリック、ドラッグという動作をおこないます。そしてマウスによる処理、例えば画面上に絵を描いたり、範囲を指定したり、メニューを選択したり、というような処理は、これらの動作をうまく組み合わせることで実現しています。
プログラムの中でこれらの動作が発生したことを知るには、マウスの動きやボタンの状態を常に検出する必要があります。そして、マウスの動きを検出したら、すぐにその変化に対応する処理をおこなうようにプログラムを作ります。
マウスを使うには、マウスを使えるようにするMOUSE 命令と、マウスの動きやボタンの状態を知るためのMOUSE 関数を使います。

**MOUSE命令**

マウスを使い始めたり、画面にマウスカーソルを表示したり、マウスの使用をやめるときには、MOUSE 命令を使います。

| MOUSE命令 | 意味 |
|---|---|
| MOUSE 0 | マウス使用開始 |
| MOUSE 1,,,1 | マウスカーソルの表示 |
| MOUSE 5 | マウス使用終了 |

MOUSE 0 とMOUSE 1,,,1 を続けて実行すると、マウスカーソルが画面に表示され、マウスが使えるようになります。マウスを使い終わる時には、MOUSE 5 を実行してマウスの使用を終了することを宣言します。マウスカーソルが画面から消去され、マウスは使用できなくなります。

## 1. マウスの処理

1

**MOUSE 関数**

マウスの動きやボタンの状態を知るためには、MOUSE 関数を使います。
MOUSE 関数を使うと、マウスカーソルの位置、マウスボタンがクリックされたかなどのマウスのあらゆる状態を、関数が返す値で知ることができます。
マウスのどの状態を調べるかは、MOUSE 関数に与える引数の値で指定します。

| MOUSE 関数 | 意味 | 関数の返す値 |
|---|---|---|
| MOUSE(0) | ﾏｳｽｶｰｿﾙ の現在位置（水平位置） | X座標の 0から639 まで |
| MOUSE(1) | ﾏｳｽｶｰｿﾙ の現在位置（垂直位置） | Y座標の 0から479 まで |
| MOUSE(2,0) | ﾏｳｽ の左ﾎﾞﾀﾝ が現在押されているか | 押す = -1　離す = 0 |
| MOUSE(2,1) | ﾏｳｽ の右ﾎﾞﾀﾝ が現在押されているか | 押す = -1　離す = 0 |
| MOUSE(3,0) | ﾏｳｽ の左ﾎﾞﾀﾝ のｸﾘｯｸ回数 | ｸﾘｯｸされた回数 |
| MOUSE(3,1) | ﾏｳｽ の右ﾎﾞﾀﾝ のｸﾘｯｸ回数 | ｸﾘｯｸされた回数 |
| MOUSE(4,0) | 左ｸﾘｯｸ時の水平位置 | X座標の 0から639 まで |
| MOUSE(5,0) | 左ｸﾘｯｸ時の垂直位置 | Y座標の 0から479 まで |

MOUSE 関数は「システム関数」のひとつです。
システム関数とは、コンピュータの内部の状態を知るための関数です。
この関数を実行すると、コンピュータ内部の状態を、関数の返す値で知ることができます。
システム関数には、マウスの他にキーボードやモデムの状態を知る関数などがあります。

## 1. マウスの処理

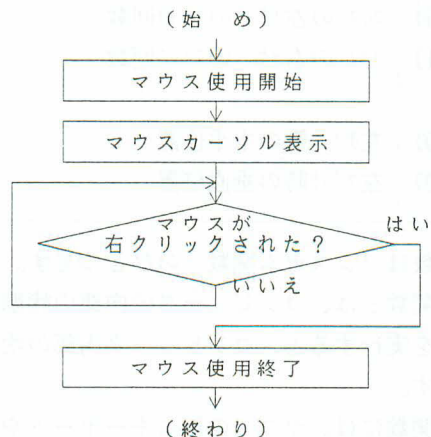
チャレンジ

● マウスボタンによる終了

マウスの右ボタンが押されるまでマウスカーソルを表示するプログラムを作ってみましょう。

このプログラムは次のような考え方で作ります。

WHILE-WEND命令を使って、マウスの右ボタンが押されない間、ただマウスカーソルを表示し続けます。右ボタンが押されたかどうかを調べるには、MOUSE（3,1）を使います。このとき、MOUSE(2,1)を使いません。MOUSE(2,1)は、関数を実行したときに、ちょうどマウスのボタンが押されていないと、ボタンが押されたかどうかを知ることはできません。それに対して MOUSE(3,1) は、関数を実行したときにボタンが押されていなくても、ボタンが押されたかどうかを知ることができます。

条件式 MOUSE(3,1) = 0 でマウスの右ボタンが一度も押されていないことがわかります。

（プログラム例3-1-1 ：マウスの右ボタンが押されるまで待つ）

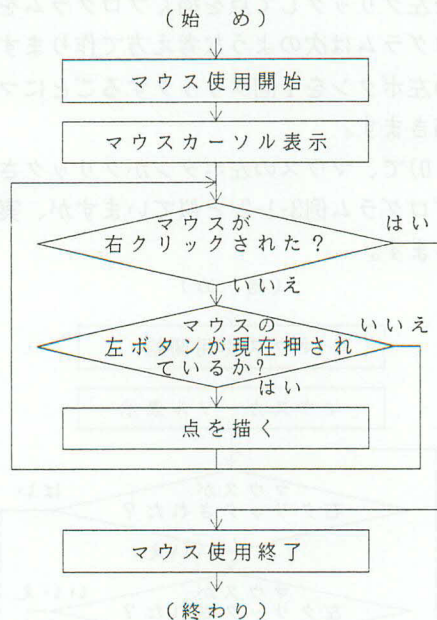
```
100 '-----------------------------------------------
110 ' マウスボタンを押すまでカーソルを表示する
120 '-----------------------------------------------
140 CLS
150 MOUSE 0
160 MOUSE 1,,,1
170 WHILE MOUSE(3,1) = 0
180 WEND
190 MOUSE 5
200 END
```

●ドラッグによる動作

マウスをドラッグして点線を描くプログラムを作ってみましょう。

このプログラムは次のような考え方で作ります。

マウスの左ボタンを押したら点を描く動作を始め、ボタンを離したら点を描く動作をやめると考えます。これはマウスボタンを押しているときと、離しているときとで動作を変えるということです。条件式 MOUSE(2,0) = -1でマウスの左ボタンが押されているかどうかがわかります。この条件式によって条件分岐をおこないます。マウスカーソルの位置は、MOUSE(0)とMOSUE(1)を使って調べます。

前のプログラム例3-1-1 の手法と組み合わせれば、マウスの右ボタンを押すと終了させることができます。

（プログラム例3-1-2 ：ドラッグして点線を描く）

```
100 '-----------------------------------------------
110 ' ドラッグして点線を描く
120 '-----------------------------------------------
130 CLS
140 MOUSE 0
150 MOUSE 1,,,1
160 WHILE MOUSE(3,1) = 0
170   IF MOUSE(2,0) = -1 THEN
180     X = MOUSE(0): Y = MOUSE(1)
190     PSET(X,Y),7
200   ENDIF
210 WEND
220 MOUSE 5
230 END
```

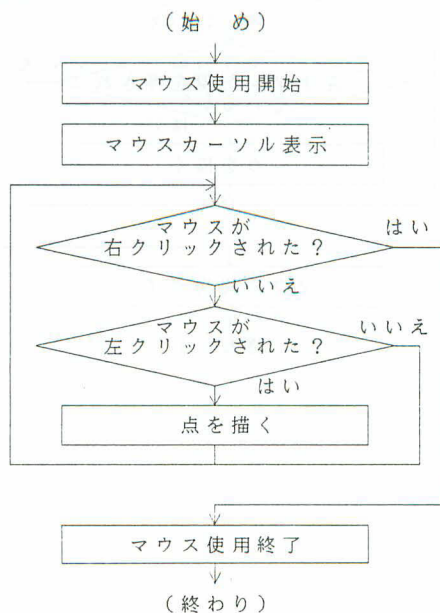
● クリックによる動作

マウスを左クリックして点を描くプログラムを作ってみましょう。

このプログラムは次のような考え方で作ります。

マウスの左ボタンを１回クリックするごとにマウスカーソルの位置を調べ、点を１つ描きます。

MOUSE(3,0)で、マウスの左ボタンがクリックされたかどうかがわかります。

☞前のプログラム例3-1-2 と似ていますが、実際に実行してみると違いがはっきりします。

（プログラム例3-1-3 ：マウスの左クリックによって点を表示する）

```
100 '---------------------------------------------------------
110 '  マウスをクリックして点を描く
120 '---------------------------------------------------------
130 '
140 CLS
150 MOUSE 0
160 MOUSE 1,,,1
170 WHILE MOUSE(3,1) = 0
180   IF MOUSE(3,0) >= 1 THEN
190     X = MOUSE(4,0): Y = MOUSE(5,0)
200     PSET(X,Y),7
210   ENDIF
220 WEND
230 MOUSE 5
240 END
```

●マウスによるメニュー選択

メニューから［表彰式］と［ブザー］のどちらかのボタンを選択して、それぞれ違う音楽を鳴らすプログラムを作ってみましょう。

このプログラムは次のような考え方で作ります。

まずマウスの左ボタンがクリックされたかどうかを調べます。クリックされたときは、そのときのマウスカーソルの座標を調べます。そしてその座標の位置が［表彰式］または［ブザー］のボタンの上かを調べ、もしそうならそのボタンに対応する音楽を鳴らします。

マウスが左クリックされたかどうかはMOUSE(3,0)で、左クリック時の座標は、MOUSE(4,0)とMOUSE(5,0)でわかります。

このプログラムは、ファイルの削除やプログラムの終了など、ユーザに確認を求める処理に応用できます。

（始　め）

マウス使用開始

マウスカーソル表示

メニュー表示

マウスが左クリックされたら、その位置を調べる

「表彰式」が左クリックされた？

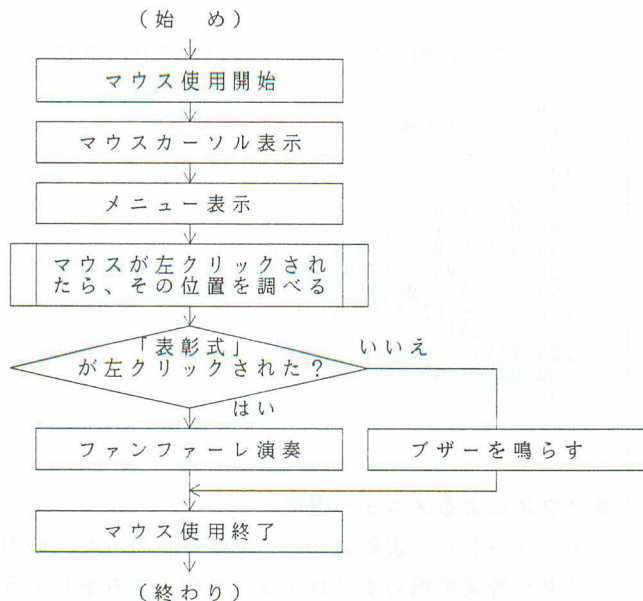
いいえ

はい

ファンファーレ演奏

ブザーを鳴らす

マウス使用終了

（終わり）

（プログラム例3-1-4 ：マウスによるメニュー選択）

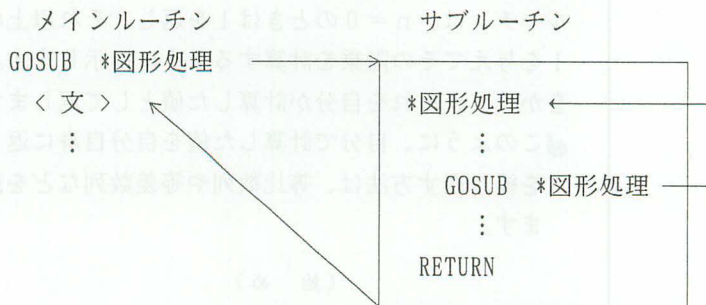
```
100 '-----------------------------------------
110 ' メニューから項目を選択する
120 '-----------------------------------------
130 CLS
140 MOUSE 0
150 MOUSE 1,0,0,1
160 GOSUB *MESSAGE
170 GOSUB *CHOOSE-YN
180 IF A$ = "Y" THEN
190   PLAY "T150L16EFGAG8G8>C4<G4F8EDD8.CC2"
210 ELSE
220   BEEP
230 ENDIF
240 MOUSE 5
250 END
260 *MESSAGE
270   LINE(0,0)-(220,80),PSET,7,B
280   SYMBOL(16,16),"どちらかを選んでください",1,1,7
290   LINE (20,40)-(80,60),PSET,7,B
300   LINE (120,40)-(180,60),PSET,7,B
310   SYMBOL(30,42),"表彰式",1,1,7
320   SYMBOL(130,42),"ブザー",1,1,7
330 RETURN
340 *CHOOSE -YN
350   F = 0
360   WHILE F = 0
370     IF MOUSE(3,0)  THEN GOSUB *CHECK-YN
380     IF A$ = "Y" OR A$ = "N" THEN F = 1
390   WEND
400 RETURN
410 '
420 *CHECK-YN
430   X = MOUSE(4,0): Y = MOUSE(5,0)
440   IF 20 <= X AND X <= 80 AND 40 <= Y AND Y <= 60 THEN A$ = "Y"
450   IF 120 <= X AND X <= 180 AND 40 <= Y AND Y <= 60 THEN A$ = "N"
460 RETURN
```

# 2. 再帰的なプログラム



プログラムを作る手法のひとつに「再帰」があります。
再帰とは、ある動作の作用が、その動作をおこなったもの自身に返ってくるようなはたらきをすることをいいます。つまり、再帰的なプログラムとは、ある処理の中でその処理自身を呼ぶプログラムのことです。
例えばひとつの図形を少しずつ位置を変えて10個描き、大きな図形にするプログラムを作るとします。図形を描く処理を10回記述する必要はありません。1回だけ記述して、その処理を位置を変えて10回繰り返して実行させればよいのです。
このようなプログラムでは、図形を描く部分をサブルーチンにして、そのサブルーチンの中で自分自身を呼ぶというような方法を使います。これを「再帰的サブルーチン」といいます。

再帰的サブルーチン

再帰的サブルーチンは、メインルーチンからGOSUB 命令で呼び出したサブルーチンの実行中に、さらにGOSUB 命令で自分自身を呼び出して実行します。
このとき、自分で自分自身を呼び出した回数は、配列変数の添字を減らして数え、呼んだ自分と、呼ばれた自分で使う変数が一致しないようにします。

## 2. 再帰的なプログラム

チャレンジ

● 再帰による計算

入力された数値の階乗を計算して、その結果を表示するプログラムを作ってみましょう。

このプログラムは次の考え方で作ります。

4の階乗は、4！で表され、結果は4×3×2×1×1＝24となります。

これを数式で定義すると、

$$n! = \begin{cases} n\times(n-1)! & \text{for } n\geqq 1 \\ 1 & \text{for } n=0 \end{cases}$$

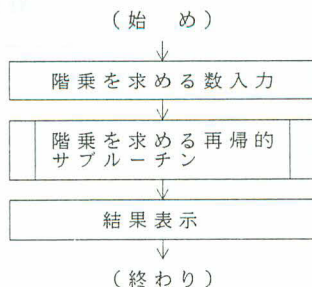
となります。そして、4！を、この数式に当てはめて計算すると、

$$4! = 4\times3! = 4\times3\times2! = 4\times3\times2\times1! = 4\times3\times2\times1\times0!$$
$$= 4\times3\times2\times1\times1$$

となります。

この式をそのままプログラミングすると長いステップが必要となります。こののようなときに再帰的サブルーチンを使います。

階乗を計算する再帰的サブルーチンに数nが与えられるとします。再帰的サブルーチンは、n＝0のときは1を返し、それ以上のときには、自分自身にn－1を与えてその階乗を計算するように指示します。そして返ってきた値とnとをかけて、それを自分が計算した値として返します。

このように、自分で計算した値を自分自身に返して、その値を基に同じ計算を繰り返す方法は、等比数列や等差数列などを計算するときにも、応用できます。

（ プログラム例3-2-1 ：与えられた数値の階乗を求める ）

```
100 '-----------------------------------------------------
110 ' 階乗を求める
120 '-----------------------------------------------------
130 DIM V(20)
140 INPUT " 階乗を求める数は？", N
150 GOSUB *CALC
160 PRINT V(N)
170 END
180 '-------------------------------------
190 *CALC                          '階乗を求める再帰的サブルーチン
200   IF N = 0 THEN
210     V(N) = 1                   ' 0! = 1 の式
220   ELSE
230     N=N-1: GOSUB *CALC: N=N+1  ' (n-1)! を V(N-1) に求める
240     V(N) = N * V(N-1)          ' n! = (n-1)!  の式
250   ENDIF
260 RETURN
```

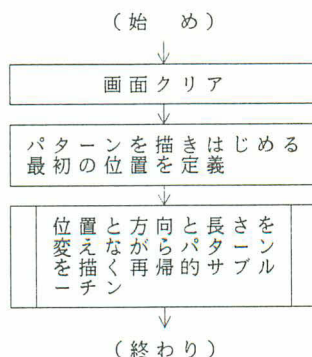
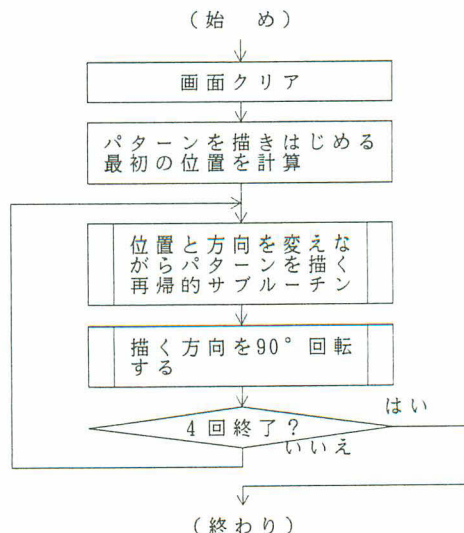
## 2. 再帰的なプログラム

● 再帰による図形

1本の幹の先端が次々と2本の枝に分かれ、大きな木を描く2分木という図形と、\/の形のパターンを方向を変えてつなぎ合わせてクローバーのような模様を描くシェルピンスキー曲線という図形を描くプログラムを作ってみましょう。

これらのプログラムは、次のような考え方で作ります。

最初に基となる図形を描き、そこから同じ図形を方向を変えてつなげて、より大きな図形に発展させます。これを実現するには、再帰的サブルーチンを使います。サブルーチンの中で、図形を描く角度を変えて自分自身を呼ぶことを繰り返します。

再帰的サブルーチンでは自分自身を呼び出す回数が終了条件になります。この回数は、配列変数の添字を減らして数えます。

（プログラム例3-2-2 ：2分木を描く）

```
100 '--------------------------------------------------
110 '  2 分 木 を 描 く
120 '--------------------------------------------------
130 '
140 DIM X(20),Y(20),K(20),A(20)
150 PI = 3.14159!
160 '
170 CLS
180 L  = 8
190 X(L-1) = 320               ' 最 初 の 枝 を 描 き 始 め る 水 平 位 置
200 Y(L-1) = 470               ' 最 初 の 枝 を 描 き 始 め る 垂 直 位 置
210 K(L-1) = 75                ' 最 初 の 枝 の 長 さ
220 A(L-1)  = 0                ' 最 初 の 枝 の 方 向
230 GOSUB *BRANCH
240 END
250 '
260 '--------------------------------------------------
270 *BRANCH                     '枝 を 描 く 再 帰 的 サ ブ ル ー チ ン
280   L = L - 1
290   IF L > 0 THEN
300     X(L-1) = X(L) + K(L) * SIN(A(L))    ' 次 の 水 平 位 置
310     Y(L-1) = Y(L) - K(L) * COS(A(L))    ' 次 の 垂 直 位 置
320     K(L-1) = .9! * K(L)                 ' 次 の 長 さ
330     LINE (X(L),Y(L)) - (X(L-1),Y(L-1)),PSET,7
340     A(L-1) = A(L) - PI/12     ' 次 の 枝 の 方 向 （ 左 の 枝 ）
350     GOSUB *BRANCH
360     A(L-1) = A(L) + PI/12     ' 次 の 枝 の 方 向 （ 右 の 枝 ）
370     GOSUB *BRANCH
380   ENDIF
390   L = L + 1
400 RETURN
```

(プログラム例3-2-3 ：シェルピンスキー曲線を描く)

```
100 '-----------------------------------------------------
110 ' シェルピンスキー曲線を描く
120 '-----------------------------------------------------
130 '
140 SCREEN @0
150 CLS
160 L = 5
170 DX  = 120 / 2 ^L: DY  = 120 / 2 ^L
180 DLX= 120 / 2 ^L: DLY =   0 / 2 ^L
190 POINT (80,0)
200 FOR I=1 TO 4
210   GOSUB *ELEMENT
220   LINE STEP(0,0) - STEP(DX,DY),PSET,7
230   GOSUB *ROT-90
240 NEXT I
250 END
260 '
270 '-----------------------------------------------------
280 *ELEMENT                         '曲線を描く再帰的サブルーチン
290   IF L <= 0 THEN RETURN
300   L = L-1
310   GOSUB *ELEMENT
320   LINE - STEP(DX,DY),PSET,7
330   GOSUB *ROT-90
340   GOSUB *ELEMENT
350   GOSUB *ROT-270
360   LINE - STEP(2*DLX,2*DLY),PSET,7
370   GOSUB *ROT-270
380   GOSUB *ELEMENT
390   LINE - STEP(DX,DY),PSET,7
400   GOSUB *ROT-90
410   GOSUB *ELEMENT
420   L = L+1
430 RETURN
440 '
450 '-----------------------------------------------------
460 *ROT-90                           ' 描く方向を９０°回転する
470   T = DX: DX = -DY: DY = T
480   T = DLX: DLX = -DLY: DLY = T
490 RETURN
500 '
510 '-----------------------------------------------------
520 *ROT-270                         ' 描く方向を２７０°回転する
530   T = DX: DX = DY: DY = -T
540   T = DLX: DLX = DLY: DLY = -T
550 RETURN
```

# 3. エラーの処理

プログラムを実行したときに、命令の使い方や表記の方法などに間違いがあるとエラーが発生し、プログラムが止まります。

F-BASIC386は、エラーが発生した行の番号と、その内容をエラーメッセージとして表示しますので、エラーメッセージにしたがってプログラムを修正します。エラーの内容は次の種類に大きく分けることができます。

- 構文上のエラー
- 構造上のエラー
- 飛び先のエラー
- 意味上のエラー

これらのエラーはプログラムに誤りがあったために発生したエラーです。しかしプログラムは正しいのに発生するエラーもあります。

このようなエラーの発生は、あらかじめ予想しておくことが大切です。そしてエラーでプログラムが中断されないようにエラー処理ルーチンを作っておきます。

**構文上のエラー**

命令のスペルや、文の表記形式を間違えたときには構文上のエラーが発生し、次のようなエラーメッセージが表示されます。

```
ＥＲＲＯＲ　２　　文法が正しくありません
ＥＲＲＯＲ　１３　変数または式の型があっていません
ＥＲＲＯＲ　２２　オペランドの記述が正しくありません
```

## 3. エラーの処理

**構造上のエラー**

構造化IF文、WHILE-WEND命令、FOR-NEXT命令、GOSUB-RETURN命令、ON ERROR GOTO-RESUME 命令の表記形式を間違えたときには、構造上のエラーが発生し、次のようなエラーメッセージが表示されます。

| | | |
|---|---|---|
| ＥＲＲＯＲ | １ | ＦＯＲ文がないのにＮＥＸＴ文がありました |
| ＥＲＲＯＲ | ２３ | ＦＯＲ文とＮＥＸＴ文の対応が正しくありません |
| ＥＲＲＯＲ | ３ | ＧＯＳＵＢ文がないのにＲＥＴＵＲＮ文がありました |
| ＥＲＲＯＲ | ２４ | ＷＨＩＬＥ文に対応するＷＥＮＤ文がありません |
| ＥＲＲＯＲ | ２５ | ＷＨＩＬＥ文がないのにＷＥＮＤ文がありました |
| ＥＲＲＯＲ | １９ | エラー処理ルーチンにＲＥＳＵＭＥ文がありません |
| ＥＲＲＯＲ | ２０ | エラー処理ルーチン以外にＲＥＳＵＭＥ文がありました |
| ＥＲＲＯＲ | ３１ | ＩＦブロックにＥＮＤＩＦ文がありません |
| ＥＲＲＯＲ | ３２ | ＩＦブロックの構造が複雑すぎます |
| ＥＲＲＯＲ | ３９ | ＩＦブロックの構成に誤りがあります |

ON ERROR GOTO-RESUME命令については☞97ページをご覧ください。

**飛び先のエラー**

GOTO命令やGOSUB命令で指定した行番号やラベルが見つからないときには、飛び先のエラーが発生し、次のようなエラーメッセージが表示されます。

| | | |
|---|---|---|
| ＥＲＲＯＲ | ２９ | 定義されていないラベル名が参照されました |
| ＥＲＲＯＲ | １２４ | 存在しない行番号が参照されています |

**意味上のエラー**

ある変数に、その変数の型で決められている範囲を超える数値を代入しようとしたときなど、変数や式の値が正しくないときには意味上のエラーが発生し、次のようなエラーメッセージが表示されます。

| | | |
|---|---|---|
| ＥＲＲＯＲ | ５ | 関数または命令文の使い方が正しくありません |
| ＥＲＲＯＲ | ９ | 配列の添え字の値が許される値を越えています |
| ＥＲＲＯＲ | １１ | ０で割ることはできません |

## 3. エラーの処理

3

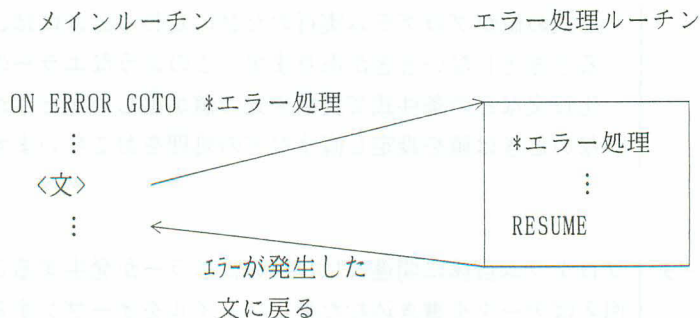
変数名のスペルミスによっても、意味上のエラーが発生します。この場合はその変数に入っている値が間違えていると解釈されているのです。

キーボードからの入力やファイルからのデータの読み込みなどによって、変数や式の値がプログラム実行のたびに変わるときには、これらのエラーは発生するときとしないときがあります。このようなエラーの発生を避けるには、構造化IF文などの条件式で変数や式の値が正しいかどうかをチェックして、正しくないときは値を設定し直すなどの処理をおこないます。

**エラー処理ルーチン**

プログラム自体に間違いはないのにエラーが発生することがあります。
例えばデータを書き込むためにファイルをオープンするとき、すでに同じ名前のファイルがあった、というようなときは、ファイル重複のエラーが発生してプログラムが止まります。
このようなときは、エラーの発生を予想して、あらかじめエラー処理ルーチンを作っておきます。
エラー処理ルーチンとは、エラーが発生したときに、プログラムの実行をとめないでエラーを処理するサブルーチンです。
ファイル重複のエラーの場合は、すでにあるファイルを消してからファイルをオープンする命令に戻る、というエラー処理ルーチンを作ります。
エラー処理ルーチンを呼ぶには、プログラムの冒頭でON ERROR GOTO 命令を実行します。これによって、プログラムの実行中に何らかのエラーが発生したとき、指定したエラー処理ルーチンが実行されます。
エラー処理ルーチンではどのようなエラーが起きたのかを判断して、処理をおこないます。エラーの内容を判断するには、システム変数ERR とERL を参照します。
システム変数ERR とERL はエラーが発生すると、自動的にセットされ、ERR にはエラー番号( エラーメッセージで表示される番号と同じもの )、ERL にはエラーの起こった行番号がセットされます。
エラー処理ルーチンからメインプログラムに戻るには、RESUME命令を使います。
RESUME命令が実行されるとメインプログラムのエラーが発生した文に戻ります。

## 3. エラーの処理

「＊エラー処理」というラベルを付けたエラー処理ルーチンは、次のような流れで実行されます。

☝システム変数とはコンピュータの内部の状態を知るためのもので、システム関数とよく似ています。しかし、システム変数はユーザがその値を自由に書き換えることができます。

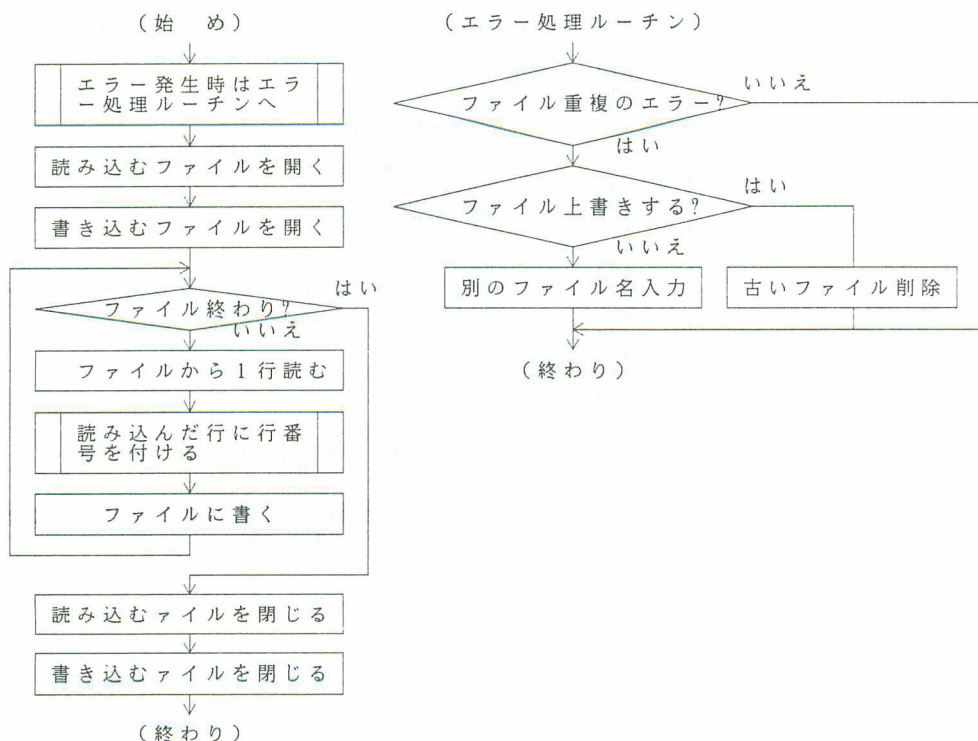
チャレンジ

● ファイル重複の場合のエラー処理

データを保存するときのファイル名入力で、すでにあるファイル名を指定したときのエラー処理ルーチンを作ってみましょう。

このプログラムは次のような考え方で作ります。

エラー処理ルーチンでは、すでにあるファイルを消してよいかどうかを確認してからファイルを消してファイルをオープンする命令に戻るようプログラムします。ファイルを消したくないときは、別のファイル名を入力して、ファイルをオープンする命令に戻るようにします。

このプログラムは、2章の「ファイルの処理」で紹介した、「プログラム例2-4-3 行番号を付けるプログラム」にエラー処理ルーチンを付け加えたものです。

# 3. エラーの処理

（プログラム例3-3-1　：行番号をふるプログラムとエラー処理）

```
100 '---------------------------------------------------
110 '  ファイルを行番号を付けて別のファイルに書き込む
120 '  （エラー処理付き）
130 '---------------------------------------------------
140 ON ERROR GOTO *OPENERROR
150 LINE INPUT "行番号を付けるファイルは？", INFILE$
160 OPEN "I", #1, INFILE$
170 LINE INPUT "書き込むファイルは？", OUTFILE$
180 OPEN "O", #2, OUTFILE$
190 L = 100
200 WHILE NOT EOF(1)
210   LINE INPUT #1, INPUT-L$
220   GOSUB *NUMBER
230   PRINT OUTPUT-L$
240   PRINT #2, OUTPUT-L$
250 WEND
260 CLOSE #1
270 END
280 '---------------------------------------------------
290 *NUMBER                 ' 読み込んだ行に行番号を付ける
300 '
310   OUTPUT-L$ = STR$(L) + " " + INPUT-L$
320   L = L + 10
330 RETURN
340 '---------------------------------------------------
350 *OPENERROR                          ' エラー処理ルーチン
360   IF ERR = 64 THEN
370     PRINT " 指定のファイルはすでに存在しています。"
380     PRINT " 上書きしますか？ (Y/N) ";
390     GOSUB *YESNO
400     IF A$ = "Y" THEN
410       KILL OUTFILE$
420       RESUME
430     ELSE
440       LINE INPUT "別のファイル名を入力してください。",OUTFILE$
450       RESUME
460     ENDIF
470   ENDIF
480 ERROR ERR
490 '---------------------------------------------------
500 *YESNO                              ' Y／N入力ルーチン
510   A$ = INPUT$(1)
520   WHILE NOT( A$ = "Y" OR A$ = "y" OR A$ = "N" OR A$ = "n" )
530     A$ = INPUT$(1)
540   WEND
550   IF A$ = "y" THEN A$ = "Y"
560   IF A$ = "n" THEN A$ = "N"
570   PRINT A$
580 RETURN
```

# 第4章

# F-BASIC386の世界を体験しよう

この章では、F-BASIC386の世界を実際に体験していただくために、いろいろなプログラム例を紹介します。どれもすぐに使えるものばかり。プログラムはすべてCD-ROMに保存されていますので、エディタに呼び出して動かすことができます。そして、本章では、これらのプログラムがどのように作られているのかを解説しています。興味のあるプログラムを実際に使いながら、自分で好きなように改良したり、実際のプログラミングに役立ててください。
●一部のプログラムでは3MBのメモリとビデオカードが必要です。

# サンプルプログラムの使い方

この章ではいろいろなサンプルプログラムを紹介します。これらのプログラムは楽しく、実用的で、今すぐ使えるものばかり。実際に使いながら、F-BASIC386の勉強に役立ててください。

これらのプログラムでは、かなり高度なテクニックも使われているので、すべてを完璧に理解することは難しいかもしれません。しかし、本格的なプログラムを作るとき、必ず参考になるはずです。

この章の説明は、必ずサンプルプログラムのリストをエディタ画面に表示して、リストを参照しながら読みすすめてください。リスト中のポイントとなる部分にはコメントが書いてありますので、処理の詳細はコメントをお読みください。

サンプルプログラムは、すべてCD-ROMの<SAMPLE>ディレクトリに入っています。サンプルプログラムをエディタ画面に呼び出す、または実行する方法は、☞ 206ページをご覧ください。また、サンプルプログラムを実行するときに、メニューで選んで実行できる便利なサンプルプログラムメニューがあります。このメニューもF-BASIC386で作られているので、他のサンプルプログラムと同じ方法でエディタ画面に呼び出したり、実行したりすることができます。

## サンプルプログラムメニューの使い方

- ディレクトリ <SAMPLE> 内に「2M_MENU.BAS」または「3M_MENU.BAS」のファイル名で入っています。

  2MB のシステムでは「2M_MENU.BAS」を、3MB 以上のシステムでは「3M_MENU.BAS」を選択します。

  2MB 用のメニューで黄色で表示されているプログラムは、3MB のシステムでなければ実行できません。
- このメニュープログラムはCD-ROM用です。サンプルプログラムをディスクにコピーして使うときは、プログラム内部のドライブ名やパス名を変更する必要があります。

サンプルプログラムメニューの画面

サンプルプログラムの実行

●サンプルプログラムを実行するには次の手順でおこないます。

①マウスカーソルを目的のプログラム名の○に合わせて左クリックします。

○が●に変わります。

②マウスカーソルを選択したプログラムの●に合わせて左クリックします。

プログラムが実行されます。

右クリックするか、他のプログラムを選択すると○に戻ります。

リストの表示

●指定したプログラムの実行を終了すると、エディタにそのプログラムのリストが残ります。

メニューの終了

●プログラムを選択せずにメニューを終了するときは、マウスカーソルを「終了」の○に合わせてダブルクリックします。

# 1. グラフィックの世界

## グラフィック機能を使う

FMTOWNS には、文字を表示するためのテキスト画面、グラフィックを表示するためのグラフィック画面、スプライトを表示するためのスプライト画面があります。このうちグラフィック画面には、16色、256 色、32,768色の３つの画面モードがあり、目的に応じて使い分けることができます。

ここでは、FMTOWNS のグラフィック機能を使った次の３つのプログラム例を紹介します。グラフィック画面やスプライト画面の使い方の参考にしてください。

- 画面をムーディーな照明にする「インテリアテレビ」。
- 自分で作ったキャラクタを画面上で自由に動かす「スプライトキャラクタアニメーター」。
- 本格的なお絵描きプログラム「386 ペイント」。

## インテリアテレビ

256 色モードで光の３原色R，G，Bの比率を連続的に変化させ、ディスプレイをインテリア照明にするプログラムです。基本となる色はデータ文としてプログラム中に書き込まれていますので、好きな色調に自由に変更することができます。

### プログラムの説明

色を連続的に変化させるには、R,G,B の値を少しずつ変えながら、PALETTE 命令を使って色を設定します。

R,G,B の値は、１～10の色調番号ごとに、それぞれ４とおりの値をDATA文で定義しておきます。この４とおりの値の間を連続的に変化させて、色を変えます。

- 色の計算は1180行から1200行でおこなっています。
- 1270行から1360行のDATA文でR,G,B のデータを定義しています。このデータをREAD命令で配列変数に読み込みます。読み込む処理は1060行から1100行の二重のFOR-NEXT命令で、DATA文の数だけ配列変数の添字を変えながらおこなっています。

### 使い方

- ディレクトリ <SAMPLE> 内に「INTERIOR.BAS」のファイル名で入っています。
- 最初のメッセージで１～10の色調番号を指定するとプログラムを実行します。
- 終了するときは、BREAK キーを押してプログラムの実行を中断します。

チャレンジ

●1060行と1100行の間のFOR-NEXT命令のループの回数を変えると、選択できる色調を変えることができます。ただし、1060行のループの回数と1270行以降のDATA文の数は必ず合わせてください。

●1270行から1360行のDATA文の色指定を書きかえると、色調を変更できます。

●1110行から1130行のWHILE-WEND命令を消し、1160行と1240行のループの外側にさらにFOR-NEXT命令で、変数NUM を1 から10まで変化させるループを追加すると、10種類の色調データすべてを順番に繰り返すことができるようになります。

●1220行と1230行の間に次のような行を追加すると、プログラムの実行中に色調を変えられるようになります。ただし、1 ～10の番号を1 ～0 の数字キーに対応させています。

```
1221  KY$ = INKEY$: IF KY$ = "" THEN *SKIP
1222    IF KY$ <= "1" AND KY$ <= "9" THEN NUM = VAL(KY$)
        ELSE IF KY$ = "0" THEN NUM = 10
1223 *SKIP
```

## 1．グラフィックの世界

### スプライトキャラクタアニメータ

Townsシステムソフトウェアの［ツール］の［スプライト編集］で作ったキャラクタをスプライトで画面に順番に表示させて動かします。あらかじめキャラクタの種類や動かし方をマウスで決定し、スタートさせます。なお、スプライトの背景は透明になりますので、ビデオカードを装備しているシステムでは、キャラクタをテレビの映像に合成することができます。

### プログラムの説明

キャラクタの一連の動きは、あらかじめ縦横32ドットの８枚のパターンで作っておきます。
キャラクタが動いているように見せるためには、８枚のパターンをスプライトパターンとして定義し、次々に表示します。スプライトキャラクタを表示したり動かしたりすることは、すべてSPRITE命令でおこないます。
キャラクタの移動はマウスで指定します。

●サブルーチン「＊スプライトパターン定義」でスプライトパターンの定義をおこなっています。
Townsシステムソフトウェアの［ツール］の［スプライト編集］で作ったパターンはF-BASIC386のプログラムのDATA文と同じかたちで保存することができます。これをREAD命令で配列変数に読み込んでから、DEFSPRITE 命令の引数でその配列変数名を指定しています。
●サブルーチン「＊入力」で、マウスによってキャラクタの動きを入力します。
●サブルーチン「＊再生」で、入力された動きを再生します。
●1990行から2200行のWHILE-WEND命令で、マウスによるメニュー選択の処理をおこなっています。1680行で定義している、FNINRANGE というユーザ定義関数が、マウスの座標が指定の範囲内にあるかどうかを調べる関数で、この関数の値によりマウスがクリックされた位置を確認します。そしてそれぞれの処理に振り分ける作業をおこないます。

**使　い　方**

●ディレクトリ <SAMPLE> 内に、

・「ANIM_CHO.BAS 」（ちょうちょのキャラクタ）

・「ANIM_PEN.BAS 」（ペンギンのキャラクタ）

のファイル名で入っています

このプログラムは、実行すると最初にスプライトパターンを配列変数に読み込む処理をおこなうので、起動に多少時間がかかります。

●終了するときは、選択モードで［終了］を左クリックします。

**●キャラクタの登録**

キャラクタはひとつずつ選択モードで指定し、入力モードでコースを決めて登録します。

●キャラクタが指定されていないと入力モードになりません。

●すでに入力されたキャラクタを指定すると動かすコースの変更ができます。

●すでに入力されたキャラクタを指定して右クリックすると登録がキャンセルできます。

**登録手順**

①１から10までのいずれかの枠を左クリックして指定します。

②画面左下の右または左向きのキャラクタを左クリックして選択します。

指定した枠内に選択したキャラクタが表示されます。

③速さ設定枠内のバーを左ボタンのドラッグで動かし、動作速度を設定します。

④［入力］を左クリックして入力モードへ移ります。

⑤左ボタンのドラッグでキャラクタを動かすコースを描きます。（☞108ページ）

⑥右クリックで選択モードに戻ります。

これでキャラクタの登録が終り、指定した枠に選択したキャラクタが残ります。

⑦必要なキャラクタの数だけ手順①～⑥の操作を繰り返します。

## 1. グラフィックの世界

●入力モードの操作

選択モードで［入力］を左クリックすると入力モードになります。このモードでキャラクタを動かすコースを決めます。キャラクタを動かすコースは、マウスをドラッグした軌跡で決まります。

- ●選択モードでキャラクタが指定されていないと入力モードに移ることができません。
- ●表示されているキャラクタはマウスで自由に移動できます。
- ●左ボタンのドラッグでキャラクタを移動させた軌跡が登録されます。
- ●入力モードに入るとすぐにマウスの軌跡が記録されます。
- ●すでに登録されているキャラクタは自動的に再生されます。
- ●左ボタンのドラッグ中に右ボタンを押すと、右ボタンが押されている間キャラクタの動きが停止します。右ボタンを離すと再びキャラクタが動きます。
- ●右クリックすると選択モードに戻ります。

●再生モードの操作

選択モードで［再生］を左クリックすると再生モードになります。このモードで登録したキャラクタの動きを再生します。

- ●左ボタンを長くクリック（3秒程度押し続ける）すると、登録されているすべてのキャラクタが同時に動き始めます。
- ●左ボタンを短くクリックすると、登録されているキャラクタを1番からひとつずつ順番に動かすことができます。途中で左ボタンを長くクリックすると残りのキャラクタを同時に動かせます。
- ●右クリックすると選択モードに戻ります。

●プログラムの終了

選択モードで［終了］を左クリックするとプログラムが終了します。

ひとこと

ビデオカードが装備されているシステムでこのプログラムを実行すると、キャラクタはビデオ画像に合成されます。自分で作ったキャラクタなどをビデオ画像に合成して録画することができます。（スーパーインポーズ☞ 137ページ）

**チャレンジ**

● 次のような操作で、自分で作ったキャラクタを動かすことができます。ぜひチャレンジしてみてください。

**①Towns システムソフトウェアの［ツール］の［スプライト編集］を使い、次のような方法でキャラクタを作ります。**

● 必ず32,768色モードでキャラクタを作ります。

● 右のような位置にキャラクタを作ります。

- 1つのキャラクタは16ドット4つの範囲を使って作ります。
- 右向き、左向き共に、2と8、3と7、4と6は同じキャラクタを使います。
- 8種類の順番さえ合っていれば2種類の違ったキャラクタを作ることができます。

| 右向き1 | 右向き2 | 右向き3 | 右向き4 |
|---|---|---|---|
| 右向き5 | 右向き6 | 右向き7 | 右向き8 |
| 左向き1 | 左向き2 | 左向き3 | 左向き4 |
| 左向き5 | 左向き6 | 左向き7 | 左向き8 |

- このプログラムで使っている「ちょうちょ」と「ペンギン」のパターンは、それぞれディレクトリ <SAMPLE> 内の「CHOUCHO.SPR」または「PENGUIN.SPR」を呼び出すと見ることができます。

**②キャラクタの作成が終わったら必ず BASICファイルでディスクに保存します。**

- 行番号は必ず 40000からにして保存します。
- 作成途中で保存するときは、必ず PATTERNファイルで保存してください。BASIC ファイルで保存すると、パターンエディタで呼び出せなくなります。

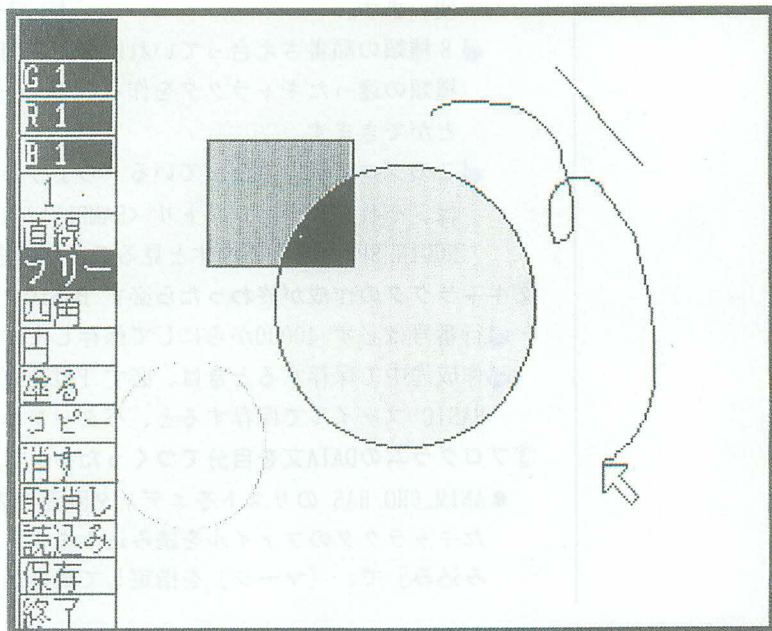
**③プログラムのDATA文を自分でつくったキャラクタに置きかえます。**

● ANIM_CHO.BAS のリストをエディタ画面に表示しおいてから、自分で作ったキャラクタのファイルを読み込みます。このとき、［ファイル］の［読み込み］で、［マージ］を指定して読み込みます。

## 1. グラフィックの世界

**386ペイント**

32,768色モードで、画面に直線や自由線、四角、円を描くことができるプログラムです。Townsシステムソフトウェア内のイラストや画像のデータを読み込んで絵を追加したり、色を付けたりすることもできます。すべての操作をマウスでおこなうことができ、ペンサイズも選べ、消しゴムやコピー、取消し（アンドゥ）の機能も付いた本格的なプログラムです。また、CD-ROM内の32,768色のイラストや「ビデオの世界」で紹介する「ビデオ映像特殊効果」プログラムや「テロップ用文字入力」プログラムで作ったファイルなどを読み込んで編集することもできます。

F-BASIC386でもここまでのプログラムを作ることができます。メニューの表示やマウスの使い方、色の決め方や色の取り込み、ディスクフルエラーの対処の仕方など高度なプログラムへの応用ができます。

**プログラムの説明**

描画処理には、直線、自由線、四角、円を描く、色を塗る、図形をコピーする、図形を消すなどの処理があります。これらのうち、直線、円を描く、図形をコピーする処理には、次のような工夫が必要です。
これらの図形は、マウスをドラッグしてマウスボタンを離したときに、はじめて図形が決定します。そのため、図形を決定するまで、つまりマウスボタンを離すまでマウスの位置は常に変わるので、その位置に対応して図形を描いて消す、ということを繰り返す必要があります。これは、XOR 演算を使ったグラフィック命令（LINE命令など）を2回繰り返すことで実現します。

- メインルーチンで、クリックされた位置がメニューの範囲内かどうかを調べます。そして、メニューの範囲内ならサブルーチン「＊エフェクトコマンド」に、範囲外ならサブルーチン「＊メインコマンド」に振り分ける処理をおこなっています。
- サブルーチン「＊エフェクトコマンド」で、メニューのどの項目がクリックされたのかを調べ、それぞれの項目に振り分ける処理をおこなっています。
- サブルーチン「＊メインコマンド」で、メニューで選択してあるモードにしたがって描画処理をおこないます。
- サブルーチン「＊直線」「＊フリー」「＊四角」「＊円」「＊塗る」「＊コピー」「＊消す」で、それぞれの描画処理をおこなっています。

**使い方**

- ディレクトリ <SAMPLE> 内に「PAINT.BAS 」（3MB用）および「PAINT_LM.BAS」（2MB用）のファイル名で入っています。
- 2MB用のプログラムでは「コピー」と「取消し」の機能が使えません。
- プログラムの実行前に32,768色のグラフィック画面が残っていると、その画面をそのままこのプログラムで扱えます。
- プログラムを終了しても、編集が終わったグラフィック画面はそのまま残ります。「ワイプ」や「ビデオ映像特殊効果」のプログラムをメニュープログラムを使わずに直接呼び出して実行すると、グラフィック画面をそのまま扱えます。
- 終了するときは、メニューの［終了］を左クリックします。

# 1．グラフィックの世界

| | |
|---|---|
| ●メニュー項目の機能と使い方 | このプログラムは、画面に表示されているメニューで描画色を決め、機能を選択して画面に絵を描きます。メニューの各項目の使い方は次のようになっています。 |
| 描画色 | **メニュー上部のR，G，Bそれぞれの枠内にあるバーをマウスで調整して描画色を決めます。枠内の数値はバーの位置を0～31の数値で表しています。**<br>●カーソルを枠内の目的の位置に合わせて左クリックすると、バーはその位置へ移動し、枠内はバーの位置に対応した色と数値に変わります。<br>●R，G，Bそれぞれの色はバーを左に動かすと暗く、右へ動かすと明るくなります。<br>●描画色はメニュー最上部の枠内に表示されます。<br>マウスを右クリックすると、カーソル位置の色を描画色にすることができます。 |
| ペンサイズ | **描画色を調整する枠の下にある数字と点が表示されている枠がペンサイズの枠です。この枠内にカーソルを合わせ、左クリックしてペンサイズを決めます。**<br>●左クリックするたびに数値と点の大きさは大きくなり、ペンのサイズは太くなります。<br>●最大9までで、さらに左クリックすると1に戻ります。 |
| 直線 | **メニュー内の［直線］を左クリックして反転表示させると、選択されている描画色、ペンサイズで直線が描けます。**<br>①直線の開始位置にカーソルを合わせます。<br>②直線の終了位置まで左ボタンを押してドラッグします。<br>③左ボタンを離します。 |
| 自由線 | **メニュー内の［フリー］を左クリックして反転表示させると、選択されている描画色、ペンサイズで自由な線が描けます。**<br>①自由線の開始位置にカーソルを合わせます。<br>②左ボタンのドラッグで、好きな自由線を描きます。<br>③左ボタンを離します。 |

四角

メニュー内の［四角］を左クリックして反転表示させると、選択されている描画色、ペンサイズで四角が描けます。

①四角のいずれかの角にカーソルを合わせます。
②対角方向へ左ボタンを押してドラッグします。
③左ボタンを離します。
右ボタンを押しながら左ボタンを離すと、四角の内側が塗りつぶされます。

円

メニュー内の［円］を左クリックして反転表示させると、選択されている描画色、ペンサイズで円が描けます。

①円の中心にカーソルを合わせます。
②円周方向へ左ボタンを押してドラッグします。
③左ボタンを離します。
右ボタンを押しながら左ボタンを離すと、円の内側が塗りつぶされます。

塗りつぶし

メニュー内の［塗る］を左クリックして反転表示させると、連続した線で囲まれた図形の内側、あるいは外側を選択されている描画色で塗りつぶせます。

①塗りつぶしたい図形の内側あるいはケイ線上にカーソルを合わせて左クリックします。
塗りつぶす図形は必ず連続した線で囲まれた図形でなければいけません。塗りつぶしは閉曲線を境界としておこなわれますので、もし、線のどこかに隙間があると、図形の内側だけでなく外側も塗りつぶされてしまいます。

コピー

メニュー内の［コピー］を左クリックして反転表示させると、指定した範囲内の図形をそのまま他の位置へコピーできます。

①コピーしたい範囲（矩形）の角にカーソルを合わせます。
②対角方向へ左ボタンを押してドラッグします。
③左ボタンを離します。
左ボタンを離すと範囲が指定され、四角枠が表示されます。
④カーソルを四角枠内に合わせます。
⑤左ボタンのドラッグで、四角枠を目的の位置まで移動します。
⑥左ボタンを離します。
右ボタンを押しながら左ボタンを離すと四角枠が残り、手順④～⑥の操作を繰り返して連続コピーができます。

# 1. グラフィックの世界

**消す**　**メニュー内の［消す］を左クリックして反転表示させせると四角の消しゴムが表示されます。消しゴムですでに描かれた図形を消すことができます。なお、消去された部分は透明色となります。**

①カーソルを消したい図形の位置へ合わせます。

②左ボタンのドラッグで、図形を消します。

左ボタンを押すとカーソル位置に四角の消しゴムが表示されます。

③左ボタンを離します。

左ボタンを離すと通常のカーソルに戻ります。

**取消し**　**メニューの［取消し］を左クリックすると、直前の操作を取消します。**

**読込み**　**メニューの［読込み］を左クリックすると、ドライブ名、ディレクトリ名、ファイル名を指定して保存されている32,768色のTIFFファイルを画面に呼び出せます。**

拡張子.TIFを付けなくても、自動的に拡張子.TIFを付けてファイルを読み込みます。

.TIF以外の拡張子を付けると、ファイル名入力のメッセージに戻ります。

ファイル名を指定せずに⏎キーだけを押すと元の画面に戻ります。

**保存**　**メニューの［保存］を左クリックすると、ドライブ名、ディレクトリ名、ファイル名を指定して画面に描かれた図形をTIFFファイルでディスクに保存します。**

拡張子.TIFを付けなくても、自動的に.TIFを付けてファイルを保存します。

.TIF以外の拡張子を付けると、ファイル名入力のメッセージに戻ります。

保存先に同じ名前のファイルがあるときは、そのファイルを消して保存するか、ファイル名を変更して保存するかを選択できます。

保存先のディスクが一杯で保存できないときは、ディスク交換のメッセージを表示します。

ファイル名を指定せずに⏎キーだけを押すと元の画面に戻ります。

**終了**　**メニューの［終了］を左クリックすると、画面の絵はそのままの状態でプログラムを終了します。**

画面の絵は「ワイプ」や「ビデオ映像特殊効果」のプログラムでそのまま使うことができます。これらのプログラムを実行するとき、メニュープログラムを実行すると、画面の絵は消えてしまうので、直接呼び出して実行します。

**チャレンジ**

●1500行のLINE命令を削除すると、メモリに残っているグラフィック画面を消さずにプログラムの実行ができるようになります。

```
1500 SCREEN@ 1: LINE(0,0)-(319,239),PSET,7,BF ←この部分削除
```

●プログラムを次のように直すと、ペンサイズに10〜12が追加され、自由線（フリー）でブラシが使えるようになります。

①2830行を次のように直します。

```
2830 PSIZE=PSIZE+1 : IF PSIZE>12 THEN PSIZE=1
```

②2870行を次のように直します。

```
2870 SYMBOL(MSX,MSY+MCH*4+2),STR$(PSIZE),.8!,.8!,0
```

③3220行から3390行を次のように変更します。

```
3220 *フリー
3221   IF PSIZE<10   THEN GOSUB  *ペン
3222   IF PSIZE>=10  THEN GOSUB  *ブラシ
3223   RETURN
3224   '
3225 *ペン
3230   GOSUB *マウス消去
3240   DEF PEN 0,PSIZE
3250   WHILE MOUSE(2,0)=-1
3260     PSET (X1,Y1),7,XOR
3270     X2=0 : Y2=0
3280     FOR I=1 TO 15
3290       X2=X2+MOUSE(0) : Y2=Y2+MOUSE(1)
3300     NEXT
3310     X2=X2/15 : Y2=Y2/15
3320     PSET (X1,Y1),7,XOR
3330     LINE (X1,Y1)-(X2,Y2),PSET,[PCG,PCR,PCB]
3340     X1=X2 : Y1=Y2
3350   WEND
3360   DEF PEN 0,1
3370   GOSUB *マウス表示
3380   RETURN
3381 '
3382 *ブラシ
3383   WHILE MOUSE(2,0)=-1
3384     X1=MOUSE(0) : Y1=MOUSE(1)
3385     X1=X1+RND(1)*(PSIZE-9)*4-(PSIZE-9)*2+1
3386     Y1=Y1+RND(1)*(PSIZE-9)*4-(PSIZE-9)*2+1
3387     PSET (X1,Y1),[PCG,PCR,PCB]
3388   WEND
3389   RETURN
3390
```

（3221〜3225：新たに追加／3230〜3380：そのまま／3381〜3389：新たに追加）

# 2. ゲームの世界

## 楽しいゲームのプログラム例

パソコンの楽しさのひとつにゲームがあります。
F-BASIC386でゲームにチャレンジしてみましょう。FMTOWNS のいろいろな機能を活用すれば、あなただけの楽しいゲームを作ることができます。
ここでは、家族みんなで楽しめる「迷路ゲーム」と「ミサイルゲーム」を紹介します。迷路を作るアルゴリズムや時間や得点のカウント、スプライトの動かし方、効果音の出し方など、ゲーム独特のテクニックを参考にしてください。

## 迷路ゲーム

**毎回違うパターンの迷路を自動的に作成し、通り抜けるまでにかかった時間を表示します。迷路を移動するキャラクタにはスプライトを使っています。**

### プログラムの説明

迷路は次のような考え方で作ります。
ひとつの迷路はたくさんの四角い部屋の集まりと考えます。それらの部屋の壁を破って道を作ります。最初に破る壁はどこでもかまいません。最初に破った壁を通って別の部屋に移動して、次の壁を破ります。これをうまく迷路ができるように繰り返していきます。部屋が全部つながったところで迷路の完成です。
部屋の状態は4ビットの2進数で次のように表します。

・すべての壁がふさがった状態 ……… 0000
・左の壁が破れた状態 ………………… 0001
・右の壁が破れた状態 ………………… 0010
・上の壁が破れた状態 ………………… 0100
・下の壁が破れた状態 ………………… 1000

壁が破れているかどうかを調べるにはAND 演算を使います。また、壁を破ったときはOR演算を使います。

● サブルーチン「＊迷路を描く」で、部屋の状態を調べながら迷路を描く処理をおこなっています。部屋の状態は配列変数MEIRO(X,Y)で表します。

**使い方**

- ディレクトリ <SAMPLE> 内に「MAZE.BAS」のファイル名で入っています。
- 迷路をクリアしたときに、表示されるメッセージで[N]キーを押すとプログラムを終了します。[Y]キーを押すと違うパターンの迷路で遊べます。

**ハードコピー**

- 迷路が完成したときに表示されるハードコピーのメッセージで[Y]キーを押すとプリンタでハードコピーを取ることができます。

**ゲームスタート**

- ハードコピーのメッセージで[N]キーを押すとゲームがスタートし、秒数がカウントされます。

**スタート位置**

- 最初にキャラクタが表示されるところがスタート位置です。

**ゴール位置**

- 迷路下側の■マークの位置がゴールです。

**キャラクタ移動**

- キーボードの[←]、[→]、[↑]、[↓]キーを使って、キャラクタを上下左右に移動します。

チャレンジ

●同じパターンの迷路を何度も繰り返しておこないたいときは、1280行のGOTO命令の行き先を1200行に変更します。IF-THEN-ELSE命令で行き先を「＊メイン」か、「1200行」かを選択できるようにすると自由に選べるようになります。

●パターンを自分で決めたいときは、1080行のRANDOMIZE 命令の後ろのTIMEを削除します。TIMEを削除すると、画面に「乱数の系列の番号を入力してください」とメッセージが表示されます。このメッセージで -2147483648から2147483647の範囲で整数を入力するとその値で迷路が作られます。同じ数値を入力すれば同じパターンの迷路が作られます。

●3140行以降のデータはスプライトのキャラクタデータです。Towns システムソフトウェアの［ツール］の［スプライト編集］で、16×16ドットのパターンを32,768色で作って入れかえると、自分で作ったキャラクタを動かすことができます。

ただし、作成するキャラクタの順番はリストの順番に合わせ、完成したキャラクタはBASIC ファイルで保存します。そして、迷路のプログラムにマージしたら、不要な部分を削除します。

自分で楽しいキャラクタを作ってみてください。

＜キャラクタの作成位置＞

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 上 1 | 上 2 | 下 1 | 下 2 | 右 1 | 右 2 | 左 1 | 左 2 |
| | | | | | | | |

## ミサイルゲーム

上空から、地上の基地をめがけてミサイルが次々と飛んできます。基地を守るためには、マウスを使ってミサイルの前方に弾幕を張ってミサイルを破壊します。最初はゆっくりとしたスピードですが、段々スピードが早くなります。基地が全部破壊されるとゲームは終了し、得点が表示されます。わりと単純ですが、遊んでみるとなかなか楽しいゲームです。高得点が出せるようになったら、自分でプログラムを修正し、スピードを早くしたり、ミサイルの数を増したりしてチャレンジしてみてください。

### プログラムの説明

基地を爆発させながらたくさんのミサイルを動かすためには、次のような処理を同時におこなう必要があります。

・ミサイルを発生させる
・弾幕を徐々に大きくする
・ミサイルが基地に当たったかどうかを判断する
・爆炎を徐々に大きくする

これらを同時におこなうには、これらの処理をすべて含んだループを作り、何回も繰り返しながらそれぞれの処理を少しずつ発展させていきます。
例えば、基地にミサイルが当たったので基地を爆発させたいときも、最初は基地にミサイルが当たったという印を付けて、爆炎を少し発生させるだけにとどめておきます。そして次の爆発の処理のときに、爆炎を前より大きくする、というようにします。

● ルーチン「＊メイン」が大きなループとなっていて、すべてのことを順番におこなっています。

### 使い方

● ディレクトリ <SAMPLE> 内に「MISSILE.BAS 」のファイル名で入っています。
● 音を出すためのファイル「EXPL.FMB」が同じディレクトリ内に必要です。

#### 弾幕を出す

● マウスカーソルをミサイルの進行方向の前方に合わせ、左クリックします。
● 弾幕は同時に9つまで出せます。

#### ゲームの終了

● 基地が全部破壊されるとゲームは終了し、得点と終了のメッセージが表示されます。
● 終了のメッセージで[Y]キーを押すとプログラムは終了します。[N]キーを押すと再度ゲームができます。

## 2. ゲームの世界

**チャレンジ**

- 同時に出現するミサイルの数は、1280行の数値で変更できます。
- ミサイルのスピードは、1300行の数値で変更できます。
- ミサイルの出現頻度は、1350行の数値で変更できます。
- 1ゲームで出現するミサイルの最大数は、1370行の数値で変更できます。
- 同時に出せる弾幕の数は、1390行の数値で変更できます。
- 2840行以降のデータはスプライトのキャラクタデータです。Towns システムソフトウェアの［ツール］の［スプライト編集］で、16×16ドットのパターンを16色で作って入れかえると、自分で作ったキャラクタでゲームを楽しむことができます。
  ただし、作成するキャラクタの順番はリストの順番に合わせ、完成したキャラクタは BASICファイルで保存します。そして、ミサイルのプログラムにマージしたら、不要な部分を削除します。
  自分でミサイルや基地のパターンを作ってみてください。

<キャラクタの作成位置>

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| ミサイルの飛行 1 | ミサイルの飛行 2 | ミサイルの飛行 3 | ミサイルの爆発 1 | ミサイルの爆発 2 | ミサイルの爆発 3 | ミサイルの爆発 4 | ミサイルの爆発 5 |
| 8 | 9 | 10 | 11 | 12 | | | |
| 弾幕の爆発 1 | 弾幕の爆発 2 | 弾幕の爆発 3 | 弾幕の爆発 4 | 弾幕の爆発 5 | | | |
| 16 | 17 | 18 | 19 | 20 | | | |
| 地上の爆発 1 | 地上の爆発 2 | 地上の爆発 3 | 地上の爆発 4 | 地上の爆発 5 | | | |
| 24 | 25 | 26 | 27 | 28 | 29 | | |
| 基地正常 | 基地正常 | 基地正常 | 基地破壊 | 基地破壊 | 基地破壊 | | |

# 3. ミュージックの世界

3

リズムマシン

**リズムマシン**

FMTOWNSは音楽を楽しむためのFM音源とPCM音源を内蔵しています。そして、このFM音源とPCM音源はF-BASIC386で自由にコントロールできます。そこで、FMTOWNSをパーカッションにするプログラムを紹介しましょう。スネアドラム、バスドラム、ハイハット、ティンパニー、シンバルなど、FM音源や PCM音源の音色を自由に組み合わせてリズムをきざむ。あなたの楽器とペアで素敵な演奏が楽しめます。

**プログラムの説明**

画面から入力したリズムパターンから、音高、音長、音程、テンポなどを指定するMML を作成して、PLAY命令で演奏します。
複数の楽器と同時に演奏するにはPLAY命令で複数のMML を指定します。作成したリズムのパターンはMML の形でファイルに保存します。

- サブルーチン「＊MML 作成」で、画面で入力されたリズムパターンから自動MMLを作成しています。
- サブルーチン「＊保存」で、作成したMML をファイルに保存しています。ここで使うWRITE 命令は、文字列を標準形式でファイルに書く命令です。

**使い方**

- ディレクトリ <SAMPLE> 内に「RHYTHM.BAS」のファイル名で入っています。
- 最初に4/4 拍子か3/4 拍子かを指定します。
  - リズム演奏モードが表示されます。
  - 楽器を指定して１小節のリズムを入力するリズム入力モードと、入力したリズムを演奏したり、テンポを変更したり、保存や読み込みをおこなう演奏モードがあります。
- リズム入力モードでは、アルファベット１文字を指定して１小節ずつリズムを指定します。
- 入力したリズムは、演奏モードでアルファベットを指定して単独でも連続でも演奏できます。
- 終了するときは、リズム演奏モードの状態でPF10キーを押します。

● 演奏モードの機能の使い方

演奏モードではキーボードのキーを使って次のような機能を使うことができます。

| | |
|---|---|
| PF2 | アルファベットを指定してリズム入力モードに移ります。 |
| ｱﾙﾌｧﾍﾞｯﾄｷｰ | 入力したリズムを演奏します。（連続指定可能） |
| 数字ｷｰ | 数字の入力で演奏時のテンポを変更します。（一分間の４分音符数を30～280 の間で変更可能） |

| | |
|---|---|
| PF 8 | ドライブ名、ディレクトリ名、ファイル名を指定して保存されているリズムを読み込むことができます。<br>☞自動的に拡張子が.RHMのファイル一覧が表示されます。ドライブ名だけを入力すると、そのドライブ内の、拡張子が.RHMのファイル一覧が表示されます。<br>☞拡張子.RHMを付けなくても、自動的に拡張子.RHMを付けてファイルを読み込みます。<br>☞.RHM以外の拡張子を付けると、ファイル名入力のメッセージに戻ります。<br>☞ファイル名を指定せずに↵キーだけを押すと元の画面に戻ります。 |
| PF 9 | 入力したリズムすべてをドライブ名、ディレクトリ名、ファイル名を指定してディスクに保存できます。<br>☞拡張子.RHMを付けなくても、自動的に拡張子.RHMを付けてファイルを保存します。<br>☞.RHM以外の拡張子を付けると、ファイル名入力のメッセージに戻ります。<br>☞保存先に同じ名前のファイルがあるときは、そのファイルを消して保存するか、ファイル名を変更して保存するかを選択できます。<br>☞ドライブ名だけを入力すると、そのドライブ内の、拡張子が.RHMのファイル一覧が表示されます。<br>☞ファイル名を指定せずに↵キーだけを押すと元の画面に戻ります。 |
| PF 10 | プログラムを終了します。 |

| | |
|---|---|
| ●リズム入力モードの機能 | リズム入力モードでは、表中の赤い入力カーソルを←、→、↑、↓キーやマウスで目的の位置へ移動してリズムを指定します。 |
| ↵ | 打点位置に●を置きます。（マウスの左ボタンも同じ機能） |
| 空白 | 打点位置の●を消します。（マウスの右ボタンも同じ機能） |
| ←、→、↑、↓ | 入力カーソルを移動します。（マウス移動でも同じ機能） |
| ｱﾙﾌｧﾍﾞｯﾄｷｰ | すでに入力したリズムを表にコピーします。 |
| PF 1 | 画面の表のリズムを演奏します。 |
| ESC | リズム演奏モードに戻ります。 |

**チャレンジ**

●1400行のPLAY命令の MMLに音量調整のコマンドVnまたは @Vnを追加すると、各楽器の音量を調整することができます。好みに合わせて調整してみてください。

●現在のプログラムはFM音源の音色を演奏しますが、プログラムを次のように変更すると PCM音源の音色を演奏できます。「F-BASIC386 V2.1 リファレンス」の PCM音源の音色番号一覧表を参考に、いろんな音色を試してみてください。

①1170行～1220行で音色番号と楽器名を PCM音源に変更します。

②1250行の LOAD@命令の後にドライブ名、ディレクトリ名、ファイル名を追加すると PCM音源のファイルを読み込んで演奏できます。そのままでは E_DRUMS.PMBの音色になります。

例）E_DRUMS.PMBの音色を演奏するときはLOAD@ "Q:¥FJ2¥TONE¥E_DRUMS.PMB" となります。

③1370行のPART命令を PART I,I+6 と書き直します。

## 3.ミュージックの世界

楽譜入力

PLAY命令などを使って音楽演奏のプログラムを作るときに、一番大変なのは MMLのコマンドの記入です。そこで、画面の五線譜にマウスを使って音符や休符などをならべていけば、自動的に MMLのデータファイルを作ってくれるという便利なプログラムを紹介します。

ソプラノ、アルト、テノール、バスの4つのパートの五線譜に自由に音符や休符、タイの記入ができます。 BASICプログラムの制約上、画面表示などに多少時間はかかりますが、直接 MMLコマンドを記入することを考えればとても便利に使えます。

**プログラムの説明**

楽譜を構成するものには音符と休符があり、音符には音程と長さが、休符には長さがあります。また、小節をまたいで2つの音符をつなげるタイもあります。

このように、データの種類が多く複雑なときは、プログラムを作るときに次のような工夫が必要です。

このプログラムでは、データを主体にしてプログラムの構造を考えます。

例えば、データを画面に表示するという処理では、音譜、休符、タイなどの各データごとに、それぞれ個別の表示ルーチンを使います。そのため、このプログラムでは、処理ごとに細かくサブルーチンに分けています。

- サブルーチン「*音符を書いて小節を更新する」で、音符を小節の中に書き入れる処理をおこなっています。その中で音符がその小節の何拍めに始まるかを覚えておいて、その位置に音符を書き入れ、書き入れる前にその位置にあった音符や休符を取り除くといった作業をおこなっています。

**使い方**

- ディレクトリ <SAMPLE> 内に「SCORE.BAS 」のファイル名で入っています。
- 起動は、音符や記号のキャラクタの読み込みのために多少時間がかかります。
- 音符や休符、タイの入力や機能の選択はすべてマウス操作でおこないます。
- [音符]、[休符]、[タイ]の入力関係のアイコンは左クリックで反転表示させるとその状態になります。
- [削除]、[挿入]、[←]、[→]、[演奏]のアイコンは左クリックで機能します。機能中だけアイコンが反転表示されます。
- [新規]、[読込]、[保存]、[MML ]、[終了]のアイコンは、左クリックで反転表示させた後、再度左クリックすると機能します。反転表示の状態で他の位置を左クリックするとキャンセルできます。

## 3. ミュージックの世界

**●楽譜の入力**　機能が選択されていないときは、常に入力できます。マウスカーソルを使って4つのパートの五線譜に自由に音符や休符、タイの入力ができます。

**記号の選択**

● ［音符］、［休符］、［タイ］のいずれかのアイコンを左クリックして反転表示し、記号の種類を決めて入力します。

［音符］、［休符］、［タイ］はどれかひとつを選択でき、反転表示されている状態でその記号が入力できます。

**入力位置**

●入力したいパートの終止線（太い縦線）の少し右側にマウスカーソルを合わせて左ボタンを押すと、その位置に音符や休符が入力できます。

すでに音符や休符が入力されているときに、終止線の左で左ボタンを押すと、最も近い音符あるいは休符の再入力になります。

**音符の上下位置**

●音符の入力では、入力位置で左ボタンを押してそのままマウスを上下にドラッグすると上下位置が変化します。

**音符や休符の長さ**

●入力位置で左ボタンを押すと、音符のときは32分音符、休符のときは32分休符が表示されます。そのままマウスを右へドラッグすると、長い音符や休符に変わります。マウスを左へ戻すと短くなります。

音符は付点音符を含め、32分音符から全音符までの10種類、休符は32分休符から全休符までの6種類が入力できます。

**変音記号の入力**

●音符の入力中に、左ボタンを押したまま、右クリックすると♯、♭、ナチュラルの変音記号が付きます。右クリックするたびに記号が変わります。

**入力の確定**

●左ボタンを離すと、そのとき表示されている音符や休符が入力されます。

音符や休符が入力されると、いったん画面を表示し直すために多少時間がかかります。

**音符や休符の変更**

●すでに入力された音符や休符にマウスカーソルを合わせて左ボタンを押すと、その位置に音符や休符の再入力ができます。

**タイの入力**

● ［タイ］を反転表示させ、マウスカーソルを同じ高さの2つの音符の間に置いて左クリックすると、その2つの音符がタイでつながれます。

すでに表示されているタイにマウスカーソルを合わせて左クリックすると、タイを消せます。

**●楽譜の演奏**　［演奏］を左クリックすると、そのとき入力されている楽譜を演奏します。

**ひとこと**

●入力された音符や休符の長さが小節の中で割り切れないときは、音符や休符は自動的に細かく分割され、音符はタイでつながれます。

**●楽譜の編集**

入力中や入力が終わった楽譜は、左右にスクロールさせて見たり、小節単位で削除や挿入することができます。
音符や休符の変更は、それぞれのアイコンを反転表示させ、直接目的の音符や休符にマウスカーソルを合わせて左ボタンを押すと再入力できます。

**スクロール**

● ←、→ のアイコンを左クリックすると、画面表示を左右にスクロールできます。

☝小節線の上に表示されている数字は小節番号です。

**小節の削除**

●削除したい小節を画面の左端に表示し、［削除］を左クリックすると、その小節は削除され、後の小節が詰められます。

**小節の挿入**

●［挿入］を左クリックすると、そのとき表示されている小節の前（左側）に新しく全休符の小節が挿入されます。

☝挿入された小節には、全休符が表示されます。休符を音符に変更しながら入力します。

**●楽譜の消去**

［新規］を左クリックで反転表示し、再度左クリックすると入力された内容をすべて消去し、第１小節の表示に戻します。
保存が終わった楽譜を消去するときや、最初から入力をし直すようなときにこの機能を使います。

**●データの読み込み**

［読込］を左クリックで反転表示し、再度左クリックすると、このプログラムで作成したデータファイルをドライブ名やディレクトリ、ファイル名を指定して読み込み、楽譜に戻すことができます。

☝このプログラムで作成され、保存されたファイルが読み込まれます。

☝機能を選択すると、カレントディレクトリにある拡張子 .SCR のファイルの一覧を表示します。

☝ファイル名の入力で、ドライブ番号だけを指定すると、そのドライブにある拡張子 .SCR のファイルの一覧を表示します。

☝拡張子の入力を省略すると、自動的に .SCR を付けてファイルを読み込みます。

☝ファイル名を入力せずに ↵ キーだけを押すと、読み込みをキャンセルし、楽譜入力の画面に戻ります。

**ひとこと**

●ディレクトリ <SAMPLE> 内に「HALLELUJ.SCR」のファイル名で簡単な楽譜の例が入っています。

## 3. ミュージックの世界

●楽譜の保存

完成した楽譜は、［保存］または［MML ］アイコンの選択で、このプログラム専用のファイルあるいは標準形式のファイルで保存できます。

専用ファイル

● ［保存］を左クリックで反転表示し、再度左クリックすると、ドライブ名やディレクトリ名、ファイル名を指定して完成した楽譜を専用ファイルに保存できます。

- 保存したファイルは、このプログラムの［読込］で楽譜に戻すことができます。
- 拡張子には自動的に .SCR が付けられます。 .SCR 以外の拡張子を入力するとファイル名入力に戻ります。
- すでに同じファイル名がディスクに保存されているときは、そのファイルを書きかえるか他のファイル名で保存するかの選択ができます。
- ドライブ名だけを入力すると、そのドライブにある拡張子.SCRのファイル一覧を表示します。
- ファイル名を入力せずに⏎キーだけを押すと、保存をキャンセルし、楽譜入力の画面に戻ります。

テキスト形式

● ［MML ］を左クリックで反転表示し、再度左クリックすると、ドライブ名やディレクトリ名、ファイル名を指定して完成した楽譜のMML データをテキスト形式のファイルとして保存できます。

- 保存したファイルは、INPUT 命令で文字列変数に読み込んで、PLAY命令で演奏させることができます。
- 拡張子には自動的に .MML が付けられます。 .MML 以外の拡張子を入力するとファイル名入力に戻ります。
- すでに同じファイル名がディスクに保存されているときは、そのファイルを書きかえるか他のファイル名で保存するかの選択ができます。
- ドライブ名だけを入力すると、そのドライブにある拡張子.SCRのファイル一覧を表示します。
- ファイル名を入力せずに⏎キーだけを押すと、保存をキャンセルし、楽譜入力の画面に戻ります。

●プログラムの終了

［終了］を左クリックで反転表示し、再度左クリックすると、プログラムを終了します。

チャレンジ

●このプログラムの［MML ］で保存したデータは、次のようなプログラムを作ると、ファイル名を指定して簡単に演奏させることができます。

```
10 OPEN "I",#1,"1:HALLELUJ.MML"← 保存されているファイルのファイル名を記入する
20 WHILE NOT EOF(1) ← ファイルの終りまで繰り返して読む
30   INPUT #1,A$,B$,C$,D$ ← 4パート分を読む
40   PLAY A$,B$,C$,D$ ← 読んだMML データを演奏する
50 WEND
60 END
```

●ディレクトリ<SAMPLE>内に「HALLELUJ.MML」のファイル名で簡単な楽譜の例が入っています。

# 4. ビデオの世界

**ビデオカードの機能をフルに引き出すプログラム例**

FMTOWNS にビデオカードを装備すると、ビデオ映像にグラフィック画面や文字、スプライトを合成したり、メモリに取り込んで特殊な処理をしたりすることができます。そして、その結果はビデオに録画できます。ここでは、次のようなプログラムを紹介します。ホームビデオの編集や画像処理に活用できます。

● ビデオ映像と32,768色のTIFFファイルのグラフィック画面をワイプする「ワイプ」
● スーパーインポーズする文字を編集する「テロップ用文字入力」
● 32,768色のTIFFファイルのグラフィック画面をビデオ映像に自由にスーパーインポーズできる「スーパーインポーズ」
● ビデオから取り込んだ映像やTIFFファイルのグラフィック画面に、いろいろな特殊効果を加える「ビデオ映像特殊効果」

☞ビデオカードはオプションです。

**ワイプ**

テレビやビデオの映像で、左右や上下方向に画面を分割しながら場面が入れかわるような処理をよく見かけます。このような処理のことをワイプといいます。
ビデオカードが装備されていると、FMTOWNS でもこのようなワイプを実現できます。ここでは、あらかじめ用意したグラフィック画面とビデオ映像を、上下左右、斜め、すだれ状などに切りかえたり、中央から窓状に広げたり、閉じたり、さらに２つの画面をブラインド状に重ねたりすることができるプログラムを紹介します。すべての機能を同時に使うこともでき、工夫しだいでいろいろなパターンを作ることができます。
ビデオの映像のほか、「３８６ペイント」で作成した画面や、これから紹介する「テロップ用文字入力」や「ビデオ画像特殊効果」で作った画面、CD-ROM内のイラストなどの32,768色のTIFFファイルを読み込んで、ビデオ映像と組合せてワイプさせることができます。ホームビデオの編集などにご利用ください。

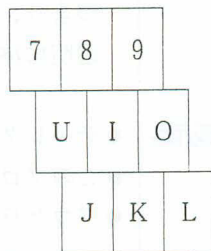
**プログラムの説明**

ワイプは、ワイプ画面とビデオ映像を重ねて、どの部分を透明にし、どの部分を不透明にするか、を指定することで実現します。透明、不透明の指定は、32,768色モードの透明ビットを操作することによっておこないます。
透明ビットを1にすると、R,G,B の指定にかかわらず、ワイプ画面は透明になり、合成されたビデオ映像が見えるというわけです。

●1850行、2340行、2500行、2590行、2760行で透明ビットの1 と0 を切りかえています。透明ビットを切りかえるには、LINE命令などのグラフィック命令でXOR演算を使います。XOR 演算は、2 回繰り返して実行すると、元に戻ります。つまり、透明だった画面は2 回XOR 演算を使ってLINE命令を実行すると、また透明に戻ります。

**使い方**

●ディレクトリ <SAMPLE> 内に「WIPE.BAS」のファイル名で入っています。
●このプログラムの実行には、ビデオカードが必要です。
●プログラムの実行前に32,768色のグラフィック画面が残っていると、その画面をそのままワイプ画面に取り込んでワイプができます。
●最初のメッセージで32,768色のTIFFファイルのファイル名を入力すると、その画面をワイプ画面に取り込んでワイプができます。
●ファイル名を入力せずに、⏎キーだけを押すとファイルの読み込みをせずにプログラムを実行します。
●終了するときは、PF10キーを押します。

| | |
|---|---|
| ●機能の使い方 | このプログラムでは、キーボードのキーおよびマウスのボタンに次のような機能が割り当てられています。 |
| 左ボタン または 右ボタン | ●ダブルクリックすると、その時点のビデオ映像をワイプ画面に取り込みます。 |
| U | ●左から右へのワイプ |
| O | ●右から左へのワイプ |
| 8 | ●上から下へのワイプ |
| K | ●下から上へのワイプ |
| 7 | ●左上から右下へのワイプ |
| J | ●左下から右上へのワイプ |
| 9 | ●右上から左下へのワイプ |
| L | ●右下から左上へのワイプ |
| I | ●ポーズ／再度押すとポーズの解除 |
| ← | ●右から左へのすだれ |
| → | ●左から右へのすだれ |
| ↑ | ●下から上へのすだれ |
| ↓ | ●上から下へのすだれ |
| PF1 | ●中央から外側へのワイプ |
| PF2 | ●外側から中央へのワイプ |
| PF3 | ●縦方向のブラインドワイプ |
| PF4 | ●横方向のブラインドワイプ |
| PF5 | ●モザイクワイプ<br>＊上記のすべてのワイプを組み合わせて同時に使うことができます。ただし、同時に使う数に比例してワイプの速度は遅くなります。 |
| PF6 | ●ワイプをキャンセルし、ビデオ映像とワイプ画面の表示を入れかえます。 |
| PF7 | ●途中でビデオ画像をワイプ画面に取り込んでも、最初に読み込んだ画面があれば、その画面をワイプ画面に復帰します。 |
| PF9 | ●ワイプを途中で固定します。PF6キーで画面を入れかえると元の画面に戻ります。 |
| PF10 | ●プログラムを終了します。 |

**テロップ用文字入力**

ビデオの映像に合成して表示する文字のことをテロップといいます。テロップは用途や背景などに合わせて、文字の種類やサイズ、色、ふちどりなど、いろいろなパターンが使われます。

ここでは画面に、書体やサイズ、色、太文字や斜体、影やふちどりなどの指定をして文字を書き、テロップ画面を作成するプログラムを紹介します。

テロップ画面は、文字以外は透明色になっていますので、次に紹介する「スーパーインポーズ」で、ビデオ映像に自由に合成することができます。

**プログラムの説明**

CD-ROMの中に入っているゴシック体、明朝体、教科書体、丸文字体の４種類の書体を利用して、画面にテロップを出しています。

- 3310行で、太文字、イタリック、影付き、アウトラインといった飾り文字を画面に表示しています。文字を表示するにはSYMBOL命令を使います。
- 3370行と3450行で、CD-ROMの中に入っている書体を1 文字ずつ読み込んでいます。

**使い方**

- ディレクトリ <SAMPLE> 内に「TELOP.BAS 」のファイル名で入っています。
- ビデオカードがなくてもプログラムは実行できますが、ビデオカードがあれば背景に映像を表示しながら作業ができます。
- このプログラムはCD-ROM内の文字パターンを扱います。プログラムの実行中は必ずF-BASIC386のCD-ROMをセットしておいてください。
- 「３８６ペイント」や「ビデオ映像特殊効果」で保存された32,768色のTIFFファイル、あるいはこのプログラムで保存されたテロップ画面を再度読み込んで文字を合成することができます。
  文字サイズ、太文字、イタリック、影付け、アウトラインは、コマンドメニューで書体に16ドット文字を指定したときのみ指定できます。他の書体では指定できません。
- 終了するときは、コマンドメニューが表示されている状態でQキーを押します。

●機能の使い方

このプログラムの各機能はコマンドメニューが表示されている状態で、最初にコマンドを入力して使います。

## 4. ビデオの世界

| | |
|---|---|
| 文字入力 | FMTOWNS の日本語機能を使ってテロップの文字が入力できます。<br>●Kキーを押し、画面のカーソル位置に文字を入力します。<br>すでに入力された文字があるときに、再度文字入力を選択すると、先に入力された文字は消去されます。<br>文字の入力方法はTownsシステムソフトウェアに添付のマニュアルをご覧ください。 |
| 文字色 | 入力された文字に 256色モードの色を指定できます。<br>●Cキーを押し、G,R,B それぞれの値(0～255)をコンマで区切って入力します。 |
| 文字位置 | テロップ文字を表示する画面上の位置が指定できます。<br>●Lキーを押し、カーソルを文字の表示開始位置（最初の文字の左上端）に合わせて左クリックします。 |
| 書体 | テロップ文字の書体にゴシック体、明朝体、教科書体、丸文字、16ドット文字が選択できます。<br>●Fキーを押し、1～5キーで希望の書体を指定します。<br>ここで、5キーを押して16ドット文字を選択すると、文字サイズや太文字、イタリック、影付け、アウトラインの指定ができます。 |
| 画面出力 | 入力したテロップ文字の指定状態を見ることができます。<br>●Oキーを押すと指定された条件のテロップ文字が表示されます。<br>●キーボードのキーをどれか押すとコマンドメニューに戻ります。 |

**保存**　完成したテロップをTIFFファイルとしてディスクに保存します。

●Wキーを押し、ドライブ名、ディレクトリ名、ファイル名を指定して保存します。

- 拡張子.TIFを付けなくても、自動的に拡張子.TIFを付けてファイルを保存します。
- .TIF以外の拡張子を付けると、ファイル名入力のメッセージに戻ります。
- 保存先に同じ名前のファイルがあると、そのファイルを消して保存するか、ファイル名を変更して保存するかを選択できます。
- 保存先のディスクが一杯で保存できないときは、ディスク交換のメッセージを表示します。
- ファイル名を指定せずに⏎キーだけを押すと元の画面に戻ります。

**読み込み**　すでに保存されているテロップや「386ペイント」、「ビデオ映像特殊効果」、CD-ROM内のイラストなどの32,768色のTIFFファイルが読み込めます。

●Rキーを押し、ドライブ名、ディレクトリ名、ファイル名を指定して読み込みます。

- 拡張子.TIFを付けなくても、自動的に拡張子.TIFを付けてファイルを読み込みます。
- .TIF以外の拡張子を付けると、ファイル名入力のメッセージに戻ります。
- ファイル名を指定せずに⏎キーだけを押すと元の画面に戻ります。

**終了**　プログラムを終了します。

●Qキーを押すと、確認のメッセージを表示し、プログラムを終了します。

- 確認のメッセージでYキーを押すと、作成したテロップを消去してプログラムを終了します。
- 確認のメッセージでNキーを押すと、コマンドメニューに戻ります。

## 4.ビデオの世界

| | |
|---|---|
| ●16ドット文字で使える機能 | 書体の指定で「16ドット文字」を指定すると、文字サイズや太文字、斜体、影付き、ふちどりの指定ができます。他の書体では指定できません。 |
| 文字サイズ | 文字の大きさを0～255の数値範囲で指定して変更できます。<br>●Pキーを押し、数値を入力します。最初の状態では16となっています。 |
| 太文字 | 文字の線の太さを大きくして強調します。<br>●Bキーを押すと指定され、「太文字」の表示が反転表示になります。再度Bキーを押すと、指定は取り消されます。 |
| イタリック | 文字を斜め（イタリック文字）にします。<br>●Iキーを押すと指定され、「イタリック」の表示が反転表示になります。再度Iキーを押すと、指定は取り消されます。 |
| 影付け | 文字に影を付けます。<br>●Sキーを押すと指定され、「影付け」の表示が反転表示になります。再度Sキーを押すと、指定は取り消されます。 |
| アウトライン | 文字の周囲に黒いふちどりを付けます。<br>●Tキーを押すと指定され、「アウトライン」の表示が反転表示になります。再度Tキーを押すと、指定は取り消されます。 |

**ひとこと**

- 入力した文字の編集は、何度でもおこなうことができます。
- 入力した文字の情報は、保存のときにグラフィック画面に合成されます。このため、いったん保存したテロップ画面の文字を変更することはできません。
- すでに保存されているテロップ画面を読み込んで、違う指定の文字を合成することができます。

**スーパーインポーズ**

ビデオ映像に画像やイラストなどを合成することを、スーパーインポーズといいます。
ここでは、「テロップ用文字入力」で作成したテロップ画面を読み込んで、ビデオ映像にスーパーインポーズするプログラムを紹介します。
「テロップ用文字入力」で作成されたテロップ画面は32,768色のTIFFファイルとして保存されています。このため、このプログラムは32,768色のTIFFファイルであれば「386ペイント」、「ビデオ映像特殊効果」で保存したファイルもCD-ROM内のイラストも読み込んでスーパーインポーズできます。
スーパーインポーズは、ビデオ映像にそのまま重ねるだけでなく、上下左右の方向や速度を指定してワイプのように重ねることができます。ホームビデオのタイトルやコメントの編集に使えます。

**プログラムの説明**

絵を高速で画面に表示するには配列変数を使います。
●サブルーチン「＊読み込み」でPUT @A 命令、GET @A 命令、LOAD@命令を使っています。
PUT @A 命令は、配列変数に取り込んだ絵を画面に表示します。
GET @A 命令は、画面に表示された絵を配列変数に取り込みます。
LOAD@命令は、TIFF形式でファイルに保存された絵を画面に表示します。
●「＊メイン」というルーチンが、キーボードから入力されたキーによって、それぞれの処理へ振り分けるという作業をおこなっています。

**使い方**

●ディレクトリ <SAMPLE> 内に「SIMPOSE.BAS 」のファイル名で入っています。
●このプログラムの実行にはビデオカードが必要です。
●プログラムの実行前に32,768色のグラフィック画面が残っているときに、メニュープログラムを使わずにこのプログラムを直接実行すると、その画面をそのままスーパーインポーズできます。メニュープログラムを使うとグラフィック画面は消去されます。
●終了するときは、コマンドメニューが表示されている状態でQキーを押します。

# 4.ビデオの世界

| | |
|---|---|
| ●機能の使い方 | このプログラムの各機能はコマンドメニューが表示されている状態で、最初にコマンドを入力して使います。 |
| 方向 | スーパーインポーズする画面をどの方向から出すかを指定できます。<br>●Dキーを押し、0～4キーで進入方向を指定します。0の「動かさない」を指定すると直接画像に重ねて表示されます。 |
| 速度 | スーパーインポーズする速度を指定できます。<br>●Sキーを押し、1～9キーで進入速度を指定します。 |
| 中央で止まる | 画面が中央にきたときにいったん停止させることができます。<br>●Tキーを押します。再度Tキーを押すと、指定は取り消されます。 |
| 読み込み | ディスクに保存されている「テロップ」や「386ペイント」、「ビデオ画像特殊効果」などの32,768色のTIFFファイルを画面に読み込みます。<br>●Rキーを押し、ドライブ名、ディレクトリ名、ファイル名を指定して読み込みます。<br>拡張子.TIFを付けなくても、自動的に拡張子.TIFを付けてファイルを読み込みます。<br>.TIF以外の拡張子を付けると、ファイル名入力のメッセージに戻ります。<br>ファイル名を指定せずに↵キーだけを押すと元の画面に戻ります。 |
| テロップ出力 | 指定された方向、速度でスーパーインポーズを実行します。<br>●Oキーを押すとコマンドメニューが消えてスタンバイ状態になります。<br>●どれかキーを押すとスーパーインポーズを開始します。<br>●スーパーインポーズ中にどれかキーを押すとポーズ状態になります。<br>再度どれかキーを押すとポーズは解除されます。<br>●スーパーインポーズが終了後どれかキーを押すとコマンドメニューに戻ります。 |
| 終了 | プログラムを終了します。<br>●Qキーを押すと、確認のメッセージを表示し、プログラムを終了します。<br>確認のメッセージでYキーを押すと、画面はそのままで終了します。<br>確認のメッセージでNキーを押すと、コマンドメニューに戻ります。 |

**ビデオ映像特殊効果**

ビデオの映像データを読み込み、RGB 演算、モノクロ変換、モザイク処理、ポスタライズ、輪郭検出、レリーフなどの処理を範囲を指定しておこない、保存することができます。特に、RGB演算では、RGB各色のレベルと変分を指定できるため、色補正やネガポジ反転などをおこなうことができます。処理に多少時間はかかりますが、一昔前ならミニコンを使っても数十分かかった処理が数分でできるのです。32ビットのFM TOWNS ならではの映像処理の妙味を存分に楽しんでください。

**プログラムの説明**

特殊効果は、R,G,B にいろいろな演算をおこなうことによって実現しています。R,G,B 演算をおこなうには、次のような方法でビデオ映像を整数型の配列変数に取り込みます。

SINPUT命令を使ってビデオ映像を画面上に取り込み、それをGET@A命令を使って整数型の配列変数に取り込みます。整数型変数の16ビットにはちょうど画面の1ドットが対応します。したがって、画面の総ドット数の大きさの配列変数を用意すれば、画面全体を取り込むことができます。PAT %がそのための配列変数です。

● サブルーチン「＊RGB 値取得」では配列データの内部表現を、R,G,B に分解して値を求める処理をしています。

配列データの内部表現は次のようになっています。整数型変数の16ビットには下位ビットから順に、B が5 ビット、R が5 ビット、G が5 ビット入っています。最後の1 ビットは使われていません。この内部表現から、R 、G 、B の値を求めることができます。

RGB の値に演算を施したあとは、それをまた内部表現にもどしてPUT @A 命令を使って画面に一行ずつ描きます。SCR %がそのための配列変数です。

このサブルーチンはいろいろなところで呼び出されています。

● 「＊輪郭検出」「＊ポスタライズ」「＊RGB 演算」「＊モザイク処理」「＊モノクロ変換」「＊レリーフ」というルーチンでは、サブルーチン「＊RGB 値取得」で求められた値に、それぞれ異なった演算をおこなって、いろいろな特殊効果を作り出しています。

**使　い　方**

● ディレクトリ <SAMPLE> 内に「SFX.BAS 」のファイル名で入っています。
● このプログラムの実行にはビデオカードが必要です。
● CD-ROM内のイラストや「３８６ペイント」で作成した絵などの32,768色のTIFFファイルを読み込んで処理することができます。
● プログラムを終了するときは、ESC キーを押します。

## 4. ビデオの世界

●機能の使い方　このプログラムの各機能は、メニュー表示の状態で番号を指定して使います。

映像の取り込み　ビデオカードに入力されている映像をメモリに取り込みます。

●[0]キーを押し、マウスを左クリックすると、その瞬間に表示されているビデオ映像を画面に読み込みます。

画像の読み込み　ディスクに保存されている32,768色のTIFFファイルを画面に読み込みます。

●[1]キーを押し、ドライブ名、ディレクトリ名、ファイル名を指定すると、保存されているTIFFファイルを画面に読み込むことができます。

- 拡張子.TIFを付けなくても、自動的に拡張子.TIFを付けてファイルを読み込みます。
- .TIF以外の拡張子を付けると、ファイル名入力のメッセージに戻ります。
- ファイル名を指定せずに[↵]キーだけを押すと元の画面に戻ります。

範囲の設定　処理の対象とする範囲の指定ができます。

●[2]キーを押し、左ボタンのドラッグで演算の対象とする範囲を指定します。

- 一度指定すると、再度指定されるまでその範囲が有効となります。
- 範囲を指定しないと、画面全体が処理の対象となります。

**ＲＧＢ演算**　Ｒ、Ｇ、Ｂそれぞれのレベルと変分を変えて画像を再表示します。

● 3 キーを押し、ＲＧＢ各色の変分とレベルを指定すると、指定されている範囲内がＲＧＢ演算された結果に変わります。

☞変分には、実際のレベルとこれから設定する基準とするレベルの差を、大きくするか小さくするかをパーセンテージで入力します。

☞レベルには、明るさのレベルではなく演算の基準とするレベルを、0から31の数値範囲で入力します。中間値は15です。

**ひとこと**　RGB各色のレベルは32段階で画面に表示されます。RGB演算では、基準とするレベルを決め、そこから実際のレベルまでの差の拡大率（変分）を％で指定することでRGB各色の明るさを変化させています。このため、たとえば変分として(100,100,100）と指定するとレベルには関係なく何も変化しません。変分として(-100,-100,-100)、レベルに(15,15,15)と指定すると、ネガポジ反転の指定となります。変分とレベルの組み合わせしだいで、オリジナルの映像の色をどのようにでも変化できます。いろんな組み合わせを実際に試してみてください。

**モノクロ変換**　カラーの映像をモノクロ（白黒）に変換して再表示します。

● 4 キーを押すと、指定されている範囲内をモノクロ(白黒)に変換します。

**モザイク**　画像を８×８ドットの四角で平均化して置きかえて再表示します。

● 5 キーを押すと、指定されている範囲内がモザイク状になります。

**ポスタライズ**　ＲＧＢそれぞれ３２階調の輝度表示を４階調に変換して再表示します。

● 6 キーを押すと、指定されている範囲内が64色表示に変換されます。

**輪郭検出**　左右のドット間で明るさの差が基準値より大きいところを白、小さいところを黒にし、輪郭を検出して再表示します。

● 7 キーを押すと、指定されている範囲内の映像の輪郭の部分が白で、それ以外の部分が黒で表示されます。

**レリーフ**　左右のドット間で明るさの差を検出し、その差を輝度として再表示します。

● 8 キーを押すと、指定されている範囲内の映像が浮き彫りのような表示になります。

## 4. ビデオの世界

画像の保存 処理が終わった画像を32,768色のTIFFファイルとしてディスクに保存します。

● 9 キーを押し、ドライブ名、ディレクトリ名、ファイル名を指定すると、画面に表示されている画像をTIFFファイルとしてディスクに保存します。

- 拡張子.TIFを付けなくても、自動的に拡張子.TIFを付けてファイルを保存します。
- .TIF以外の拡張子を付けると、ファイル名入力のメッセージに戻ります。
- 保存先に同じ名前のファイルがあると、そのファイルを消して保存するか、ファイル名を変更して保存するかを選択できます。
- 保存先のディスクが一杯で保存できないときは、ディスク交換のメッセージを表示します。
- ファイル名を指定せずに ↵ キーだけを押すと元の画面に戻ります。

終了 プログラムを終了します。

● ESC キーを押すと、確認のメッセージを表示し、プログラムを終了します。

- 確認のメッセージで Y キーを押すと、画面の映像はそのままでプログラムを終了します。
- 確認のメッセージで N キーを押すと、メニューに戻ります。

ひとこと

● メニューは、［空白］キーを押すと消すことができます。演算の結果をゆっくりと見たいときなどに押してください。キーボードのキーを何か押すと、メニューは再び表示されます。

● 設定されている範囲は、メニューを消すと白い四角枠で確認できます。

● 範囲の設定をなにもしないと、画面全体が演算の対象となり、処理に時間がかかります。

# 5. パーソナルオートメーションの世界

NIFTY-Serve 自動ダウンロード

今人気のパソコン通信。あなたのメッセージに、年令や職業を超えていろんな人が応えてくれる。いろんなテーマでシリアスに議論する。プライベートなメールを送る。いろんな分野の最新情報を入手する。時間も空間も超えた新しいカタチのコミュニケーション。あなたもFMTOWNS でパソコン通信にチャレンジしてみませんか。
でも、おもしろいからといって夢中になると、気になるのが時間と通信費。合理的な運用が経済的な通信のコツです。そこで、留守中でも深夜でもパソコン通信をアクセスし、メッセージをダウンロードできる、タイマー付きのオートダウンロードプログラムを紹介しましょう。

- このプログラムの実行には、ニフティ㈱が運営している有料パソコン通信ネットワークNIFTY-Serve のＩＤとパスワードが必要です。
- オプションのモデムカード、あるいは市販のモデムが必要です。

1992年11月現在のNIFTY-Serve のシステムで有効です。システムが変更された場合、そのままでは使えなくなることがあります。

このプログラムは、NIFTY-Serve で次のようなことができます。

- センターあるいはアクセスポイント(FENICS)経由でのオートログイン。
- ターミナルモードでのマニュアル操作、ダウンロード、アップロード。
- タイマー動作によるフォーラム会議室未読メッセージの自動ダウンロード。

ひとこと

- NIFTY-Serve に入会を申し込むとＩＤとパスワードがもらえます。
- NIFTY-Serve への入会申し込み方法については、ニフティ㈱へ直接お問い合わせください。

  ☎ 0120-22-1200(無料電話) 月曜～金曜 9:00 ～ 19:00 (土曜17:50 まで、日曜祭日不可)
- パソコン通信には、別売のモデムが必要です。詳しくはFMTOWNS 本体に添付のマニュアルをご覧ください。

5 NIFTY-Serve自動ダウンロード

## 5．パーソナルオートメーションの世界

**プログラムの説明**

このプログラムは、通信をおこなうため次のような特殊な処理をおこなっています。

通信にはモデムが必要です。モデムを制御するためにはATコマンドを使います。ATコマンドには、モデムを初期化したり、モデムに通信速度を設定したり、モデムを使って電話をかけるものなどがあります。

通信でデータのやりとりをするとき、FMTOWNS ではRS-232C 通信ポートを使います。RS-232C 通信ポートとは、モデムにデータを送信したり、モデムからデータを受信するための窓口のようなものです。RS-232C 通信ポートは通常のファイルと同様にオープンし、データを読み書きすることで、データの受信、送信をおこないます。

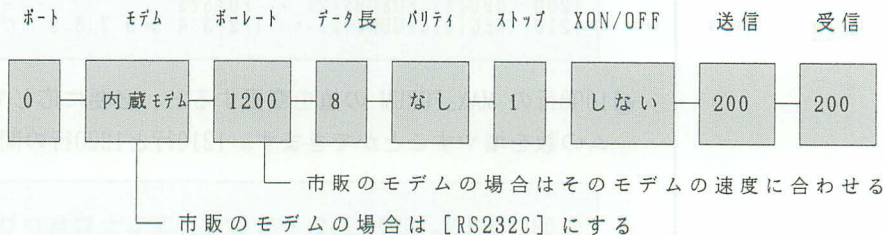
データを受信するには、割り込みルーチンを使います。データを受信したときにおこなう処理をサブルーチンにして、ON COM GOSUB命令で定義します。割り込みが起きては困るところでは、その処理の最初にCOM (0) STOP文で割り込みを禁止し、その処理の最後でCOM (0) ON文で割り込みができるようにします。

- 3520行～3640行で、いろいろなATコマンドを設定し、4030行でモデムにATコマンドを発行しています。
- 1390、1400行でRS-232C 通信ポートをオープンしています。このとき、ファイル番号といっしょに装置名RS-232C を示す"COM0:" というファイルディスクリプタを指定します。RS-232C 通信ポートは入力用と出力用とで、2つオープンします。
- サブルーチン「＊アップロード」「＊ダウンロード」の、ファイル名を入力する部分で、COM (0) STOP文とCOM (0) ON文が使われています。
- 割り込みサブルーチン「＊1 文字受信」または「＊1 文字受け取って表示」によって通信ポートからの入力を処理しています。入力の処理にはON COM (0) GOSUB 命令を使います。
- 割り込みサブルーチンは「＊1 秒カウント」で待ち時間を管理します。ON TIME GOSUB命令を使って一秒ごとのタイマ割り込みをかけています。

RS－232Cとは、FMTOWNS とモデムをつなげるための規格で、8ビットのデータをやりとりするための仕組みです。
内蔵モデムは、自動的にこの規格が満たされるように設計されています。外付けモデムは、RS－232Cケーブルを使ってつなぎます。

使　い　方

●ディレクトリ <SAMPLE> 内に「AUTOTERM.BAS」のファイル名で入っています。
●オプションのモデムカード、あるいはモデムが必要です。
●使用するモデムに合わせて、通信回線の設定が必要です。
通信回線の設定は、Townsシステムソフトウェアの［設定］の［通信回線設定］でおこないます。

＜オプションのモデムカードのときの設定例＞

●自動ダウンロード中に何か不都合な状態が起こると、自動的に回線を切ってプログラムを終了します。
●プログラム中に入力する電話番号を間違えないように気を付けてください。
●プログラム中の再ダイアルの間隔は、電気通信事業法の端末設備規則により最低50秒以上、回数は2回までに決められています。

使用前の準備

このプログラムは、実行の前に、アクセスポイントの電話番号と、あなたの ID 番号をプログラム中に入力し、ディスクへ保存してからお使いください。
なお、センターの電話番号で直接アクセスするときは、アクセスポイントの電話番号を入力する必要はありません。
●アクセスポイント(FENICS)経由で使用するときは、1120行の先頭に ' を付け、1110行の先頭の ' を削除し、電話番号を入力します。
●1140行にあなたの ID 番号を入力します。

```
1110 'NET_TELNUM$     = "XXX-XX-XXXX":FENICS=TRUE
1120  NET_TELNUM$     = "03-730-4611":FENICS=FALSE
1130  CONNECTION_ID$= "SVC/SJIS"
1140  USER_ID$        = "XXXXXXXX"
```

ID番号（↑）

← アクセスポイントを使うときは電話番号を入力し ' を外す

← 頭に ' を付けるか削除する

5

NIFTY-Serve自動ダウンロード

●1180行から1210行の間に、自動ダウンロードするフォーラム名と会議室番号を登録します。

☞すでにFFMT（ TOWNSフォーラム）とFOASYS（オアシスフォーラム）が登録されていますが、必要に応じて変更してください。

```
1180  REGIST_FORUM$(1) = "FFMT"                 フォーラム名
1190  REGIST_ROOM$(1) = "1 2 3 4 5 6 7 910"     会議室番号（ダウンロ
1200  REGIST_FORUM$(2) = "FOASYS"               ードする会議室の番号
1210  REGIST_ROOM$(2) = "1 2 3 4 5 6 7 8 9"     だけを記入する）
```

☞1090行の MAX_FORUM の値を変更すると、必要に応じて登録できるフォーラムの数を増やすことができます。1210行と1220行の間に追加します。

```
1090  MAX_FORUM = 2  ←ここで指定した数値の数だけ登録できる
```

< FFM3(FMフォーラム3「ホビー館」）の会議室6と8、9を登録した例 >

```
1090  MAX_FORUM = 3
```

```
1180  REGIST_FORUM$(1) = "FFMT"                 この間に1180行から
1190  REGIST_ROOM$(1)  = "1 2 3 4 5 6 7 910"    1210行と同じような
1200  REGIST_FORUM$(2) = "FOASYS"               プログラム文を追加
1210  REGIST_ROOM$(2)  = " 1 2 3 4 5 6 7 8 9"   する
1211  REGIST_FORUM$(3) = "FFM3"                 ただし、( ) 内の数
1212  REGIST_ROOM$(3)  = " 6 8 9"               字は連番にする
1220  CLOCK = TRUE
```

**自動ダウンロードの手順**

使用前の準備が終わったらプログラムを実行し、次のような操作で自動ダウンロードをおこないます。

①「自動ダウンロードしますか」のメッセージで、Yを入力します。

②「パスワードを入力してください」のメッセージで、あなたがNIFTY-Serveに登録しているパスワードを入力します。

③「フォーラム名を入力してください」のメッセージで、ダウンロードしたいフォーラムの略号を入力します。

- ⏎キーだけを押して入力を省略するとTOWNSフォーラム[FFMT]になります。
- ［例］TOWNSフォーラムではFFMT 、オアシスフォーラムではFOASYSでです。

④「ファイル名を入力してください」のメッセージで、ドライブ名、ディレクトリ、ファイル名を入力してダウンロード先を指定します。

- ファイル名には自動的に .NET の拡張子が付けられます。
- すでに同じ名前で保存されているファイルがあるときは、そのファイルに上書きすることもできます。

⑤「ログインの予約時刻を入力してください」のメッセージで、プログラムの実行を開始する時刻を入力します。

- 午前零時が 00:00:00 、正午が 12:00:00 、午後6時が18:00:00 のように24時間制で、必ず秒数まで入力します。
- ⏎キーだけを押して入力を省略すると、すぐにプログラムを実行します。
- 画面に予約したフォーラム名、ダウンロードファイル名、現在時刻と予約時刻が表示され、設定された時刻になると自動ダウンロードをおこない、作業が終了するとプログラムを終了します。

**マニュアル操作の手順**

使用前の準備が終わったらプログラムを実行し、次のような操作でターミナルモードに入ります。ターミナルモードでは、マニュアル操作によるNIFTY-Serve のアクセスと、ダウンロード、アップロードができます。

①「自動ダウンロードしますか」のメッセージで、Nキーを押します。

②「NIFTY にオートログインしますか」のメッセージでYまたはNキーを押します。

- Nキーを押すとターミナルモードになり、マニュアル操作ができます。
- Yキーを押すと「パスワードを入力してください」のメッセージが表示されます。登録しているパスワードを入力すると、自動的にNIFTY-Serve へ接続します。NIFTY-Serve へ接続後はマニュアルでのアクセスができます。

**ターミナルモードの機能**

ターミナルモードで使用中には、次のような機能が使えます。

● PF 1 キーを押すと、ドライブ名、ディレクトリ名、ファイル名を指定してディスクにダウンロードできます。拡張子には自動的に .NET が付けられます。

ファイル名の入力を省略し、⏎キーだけを押すとダウンロードの操作をキャンセルし、ターミナルモードの状態に戻ります。

再度PF 1キーを押すと、ダウンロードを終了します。

● PF 2 キーを押すと、ドライブ名、ディレクトリ名、ファイル名を指定して、ディスクに用意されている拡張子 .NET のファイルをアップロードできます。

ファイル名の入力を省略し、⏎キーだけを押すとアップロードの操作をキャンセルし、ターミナルモードの状態に戻ります。

● PF 10 キーを押すと、回線を切断した後プログラムを終了します。

**ご注意**

● モデムへコマンドを送ってもモデムから応答がないとときは、BREAK キーを何度も押し続けてプログラムを中断し、モデムの設定を確認してください。

● 万一回線に接続中にコマンドが送れなくなってしまったようなときは、次のような方法で回線を切ってください。

①BREAK キーでプログラムを中断する。

②［一行実行］でEND 命令を実行する。

③再度プログラムを実行し、ターミナルモードにする。

④PF 10キーを押して、回線を切る。

PF 10キーを押しても回線が切れないときは、モデムからケーブルを抜くかモデムの電源を切って、回線を強制的に切ってください。

チャレンジ

●次のような命令を追加すると、ターミナルモードでアクセスしている状態から PF3 キーを押すことで、フォーラム名、ダウンロードファイル名を指定して自動ダウンロードすることができます。

①1360行と1370行の間に次のようなプログラム文を追加します。

```
1356  ON KEY(3)  GOSUB  * 自動ダウンロード移行
```

②2050行と2060行の間に次のようなプログラム文を追加します。

```
2055  KEY 3, "AutoDL"
```

③2060行を次のように書き直します。

```
2060  KEY(1) ON:  KEY(2) ON:  KEY(3) ON:  KEY(10) ON
                             └追加する
```

④2120行を次のように書き直します。

```
2120  WHILE KY$ = "" AND MAINLOOP AND NOT AUTO_DOWNLOAD
                                  └追加する
```

⑤2150行と2160行の間に次のようなプログラム文を追加します。

```
2155  IF AUTO _DOWNLOAD THEN GOSUB * ﾏﾆｭｱﾙﾓｰﾄﾞの自動ﾀﾞｳﾝﾛｰﾄﾞ
```

⑥6780行以降に次のようなサブルーチンを追加します。

```
6780  '
6790  * 自動ダウンロード移行
6800      AUTO_DOWNLOAD = TRUE
6810  RETURN
6820  '
6830  * マニュアルモードの自動ダウンロード
6840    COM (0) STOP
6850    LINE(CARETX*8,CARETY*19)-(CARETX*8,CARETY*19+18),PSET,0
6860    PRINT
6870    PRINT "ﾌｫｰﾗﾑ名を入力してください（省略  現在いるﾌｫｰﾗﾑ)";
6880    INPUT FORUM _NAME$
6890    FORUM _NUM = MAX _FORUM
```

```
6900    WHILE FORUM _NUM >= 0 AND REGIST _FORUM$(FORUM_NUM) <>
        FORUM _NAME$
6910    FORUM _NUM = FORUM _NUM - 1
6920    WEND
6930    COM (0) ON
6940    IF NOT DOWN _LOAD THEN GOSUB * ダウンロード
6950    COM (0) STOP
6960    IF NOT DOWN _LOAD THEN  *ENDIF _MADL
6970      PRINT  "自動ダウンロードを開始します。"
6980      CSRX =  POS(0) : CSRY = CSRLIN
6990      CARETX = POS(0) : CARETY = CSRLIN
7000      KEY 1,"": KEY 2,"": KEY 3,"": KEY 10,""
7010      KEY(1) STOP :KEY(2) STOP :KEY(3) STOP :KEY(10) STOP
7020      ON COM(0) GOSUB  *1 文字受信
7030      COM (0) ON
7040      GOSUB  *自動ダウンロードサブ
7050      COM (0) STOP
7060      ON COM(0) GOSUB  *1 文字受KE取って表示
7070      KEY1,"DownLD":KEY2,"UpLoad":KEY3,"AutoDL":KEY10,"End"
7080      KEY(1) ON :KEY(2) ON :KEY(3) ON :KEY(10) ON
7090    *ENDIF__MADL
7100    AUTO__DOWNLOAD = FALSE
7110    CSRX = POS(0): CSRY = CSRLIN
7120    CARETX = POS(0): CARETY = CSRLIN
7130    LINE(CARETX*8,CARETY*19) - (CARETX*8,CARETY*19+18),PSET,2
7140    COM (0) ON
7150  RETURN
```

**ご　注　意**

●このプログラムの改造や、プログラム文やサブルーチンの追加は間違えないように慎重におこなってください。もし、間違ったままで通信をおこなうと、正常にアクセスできなくなることがあります。

**音声プログラムチェッカ**

プログラムを入力して最後にリストを見直すとき、誰かにリストを読んでもらったら、間違いも見つけやすくなります。
そこで、FMTOWNS の PCM音源の機能を利用して、FMTOWNS がリストを読み上げてくれるプログラムを紹介しましょう。なお、このプログラムで使う音声は、Towns SOUND でサンプリングし、編集しました。

☞TownsSOUNDはオプションです。

**プログラムの説明**

BASIC のリストは、行番号、命令や関数、変数や記号などで構成され、通常のファイルと同じく文字の列として保存されています。このプログラムでは、BASIC のリストを「単語」に分け、それに対応する音声データを再生します。
ファイルの読み込みは行単位でおこない、それを文字列変数に読み込み、単語に分けて音声データを再生します。単語とは、「IF」や「PRINT」といった予約語や、「+」や「@」といった記号、変数名などのことです。

- サブルーチン「*字句解析」で、文字列変数に読み込んだデータを単語に分けます。文字列変数X$に字句解析をおこなう文字列を取り込み、変数PTの値が示す位置から字句解析を始め、変数Y に単語の種類、文字列変数Y$に切り分けた単語を返します。
- 5430行から6100行までのPCM PLAY命令で、あらかじめ録音した音声データの再生をおこないます。
  単語がよく使う予約語であれば、用意してある音声データを再生します。例えば、「IF」は「イフ」と読みます。あまり使わない予約語や、変数名やラベル名は、スペルのアルファベットを順に読むことで代用します。例えば、「MOUSE」は、「エム・オー・ユー・エス・イー」と読みます。
  数値は、位取りをあらわすために、数字の読みに、「千」や「百」といったことばを挿入して発声します。そして、先行する文字の発音によって、「千」や「百」の読み方を変えています。例えば、「２００」「３００」「８００」では、「ひゃく」「びゃく」「ぴゃく」というように、「百」を３通りに読み分けます。
- これらの手順全体をまとめているのが「*リストを読む」です。

☞ここで用いている字句解析は、BASICコンパイラがBASICプログラムを読むのと同じ仕組みです。

## 5. パーソナルオートメーションの世界

**使い方**

- ディレクトリ <SAMPLE> 内に「PRGCHECK.BAS」のファイル名で入っています。
- このプログラムの実行には、3MBのメモリが必要です。
- プログラムを実行すると、最初に106個の音声データファイルを読み込みます。音声データファイルの読み込みには多少時間がかかります。

**ファイルを読み込む**

- アスキーセーブされたプログラムだけを読み上げることができます。
- 読み上げるファイル名の入力でドライブ名、ディレクトリ名が指定できます。
- ドライブ名だけを指定すると、そのドライブに保存されているBASICプログラム（拡張子.BAS）のファイル一覧を表示します。

**ファイルを読み上げる**

- 一度読み上げを実行した後、同じファイルを何度でも読み上げさせることができます。
- ファイルの読み上げを一時的に中断するときは［空白］キーを押します。再度［空白］キーを押すと読み上げを再開します。
- ファイルの読み上げを途中で中止したいときはESCキーを押します。

**プログラムを終了する**

- ファイル名の入力で何も入力せずに⏎キーだけを押すとプログラムを終了できます。

# 6. ビジネスの世界

TOWNSカルク

ビジネスの世界で最も人気のあるソフトのひとつに、表計算ソフトがあります。経理のデータ、売上のデータ、在庫のデータ、顧客のデータ等、ビジネスで扱うデータのほとんどが表で扱うことができます。

F-BASIC386で本格的な表計算にチャレンジしてみましょう。数値も文字も入力でき、各種計算と再計算などができます。市販の表計算ソフトと比べればまだまだ機能不足かもしれませんが、もっといろいろな機能がほしい人は、このガイドで紹介しているさまざまなプログラムを参考に、自分で機能アップしてみてください。

このプログラムでは、かなり高度なテクニックを使っていますが、これから本格的にプログラミングの勉強をしたいと考えている人には将来必ず役に立つはずです。

プログラムの説明

表計算のプログラムでは、ひとつのセルを書きかえたとき、そのセルに関係する他のセルも自動的に計算する必要があります。

例えば、あるセルの式を書きかえたとき、そのセルの式が、他のセルの式を参照している場合は、参照されたセルが計算済みでなければなりません。そうでなければもとのセルの計算を正しくおこなうことはできません。

このような処理をおこなうために、再帰などの他に「リスト処理」というプログラミングのテクニックを使っています。

リスト処理とは、ある処理をおこなうと、別の処理に影響を与えるようなときに、それぞれの処理に関連づけて変数を与えることをいいます。そして、リストと呼ばれる、線状に接続された一連の変数を操作します。

計算式を、四則演算の優先順位に従って正しく計算するには、「再帰降下法」という方法を用いています。再帰降下法とは、1 つの演算に対応して、それを計算する1 つの再帰的サブルーチンを用意するものです。

- 26580 行～26990 行で、あるセルが書きかえられたときに、そのセルを参照しているセルを書きかえる処理をおこなっています。ここで、リスト処理が使われています。
- サブルーチン「*EDIT」が、キーボードから入力されたキーによって、それぞれの処理へ振り分ける、という作業をおこなっています。

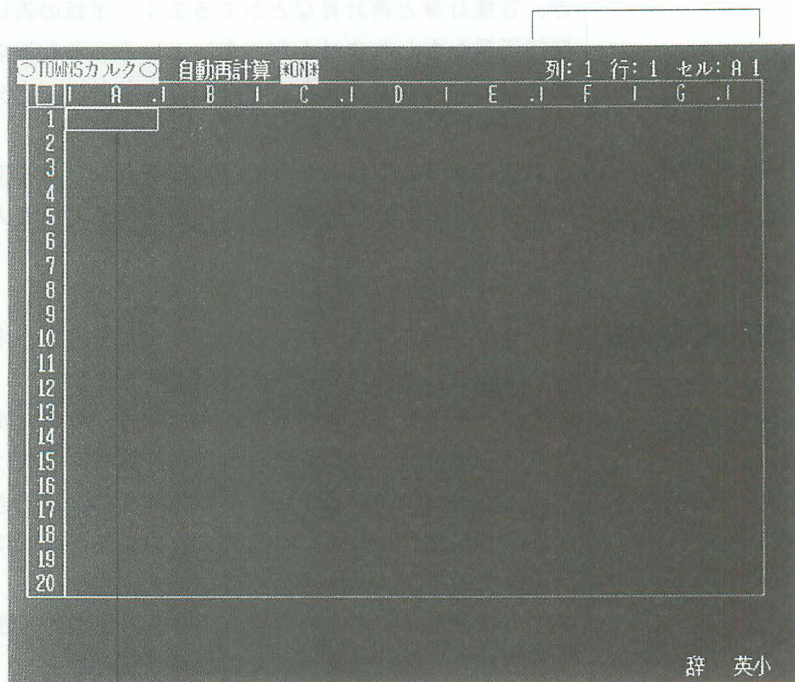
再帰降下法は、BASIC コンパイラでも使われている方法です。

## 6．ビジネスの世界

**使い方**

●ディレクトリ <SAMPLE> 内に「CALC.BAS」のファイル名で入っています。

**●TOWNSカルクの画面**

TOWNSカルクの画面は、次のようになっています。

●データの入力の機能の使い方

TOWNSカルクの表のセルには、次のような方法で数値や文字のデータと計算式が入力できます。そして、セルに計算式が入力されると、自動再計算機能により自動的に計算がおこなわれ、該当するセルの結果を書きかえます。

**数値の入力**

●数字キーを押すと、入力行に［数値］と表示され、入力されたデータは数値として扱われます。ただし、－（マイナス）は数値の符号とみなされます。

入力したデータが桁数をオーバーすると、表示上では「＊」が表示されますが、データは入力されています。列の桁数を大きくすると正しく表示されます。

**文字の入力**

●数字以外のキーが押されると、入力行に［文字］と表示され、入力されるデータは文字データとして扱われます。

入力したデータが桁数をオーバーしても、入力されたデータはすべて右側のセルにはみ出して表示されます。ただし、右側のセルに何か入力されると、はみ出したデータはそのセルより右側には表示されません。

文字はアルファベット、ひらがな、漢字が入力できます。

**数式の入力**

●@キーを押すと、入力行に［式］と表示され、セルポインタの位置に数式の入力ができます。（数式の入力の機能の使い方 157ページ）

**↵キー**

●データ入力の完了を意味し、入力行の内容をセルに移し、セルポインタを右のセルへ移動します。

何も入力せずに↵キーだけを押すと、セルの内容はそのままでセルポインタを右のセルへ移動します。ただし、列に折返しの設定がされていると、セルポインタは次の行の先頭へ移動します。折返しの設定がされた2つの列の間では、その2つの列の間を折り返しながら下へ移動します。

**ESCキー**

●入力のキャンセルを意味し、入力行の内容はセルに移されずに消去されます。

**空白キー**

●すでに入力されているデータを消去します。

**HOMEキー**

●セルポインタをA列の1行目へ戻します。

**セルポインタの移動**

- ●[←]または[CTRL]＋[A]キーで、セルポインタを左へ移動します。
- ●[→]または[CTRL]＋[F]キーで、セルポインタを右へ移動します。
- ●[↑]または[CTRL]＋[E]キーで、セルポインタを上へ移動します。
- ●[↓]または[CTRL]＋[X]キーで、セルポインタを下へ移動します。

**入力行のカーソル移動**

- ●[CTRL]＋[S]キーで、カーソルを入力中のデータの範囲で1文字左へ移動します。
  ☞セルポインタをすでにデータが入力されているセルへ移動し、この操作をすると入力モードとなり、入力行にセルのデータが表示されます。カーソルはデータの最後に表示されます。
- ●[CTRL]＋[D]キーで、カーソルを入力中のデータの範囲で1文字右へ移動します。
  ☞セルポインタをすでにデータが入力されているセルへ移動し、この操作をすると入力モードとなり、入力行にセルのデータが表示されます。カーソルはデータの先頭に表示されます。

**入力行の文字の削除**

- ●［後退］または[CTRL]＋[G]キーで、カーソル直後の1文字を削除します。
  ☞セルポインタをすでにデータが入力されているセルへ移動し、この操作をすると入力モードとなり、入力行には最初の1文字が削除されたセルのデータが表示されます。カーソルはデータの先頭に表示されます。
- ●［削除］または[CTRL]＋[H]キーで、カーソル直前の1文字を削除します。
  ☞セルポインタをすでにデータが入力されているセルへ移動し、この操作をすると入力モードとなり、入力行に最後の1文字が削除されたセルのデータが表示されます。カーソルはデータの最後に表示されます。

| | |
|---|---|
| ●数式の入力の機能の使い方 | @キー（SHIFT＋0）を押すと、入力行に［式］と表示され、セルポインタの位置に数式の入力ができます。入力できる式は次のようになります。 |
| 式の最小単位 | ●式の最小単位は数値とセル参照です。<br>数値は0から9までの数字を並べて書きます。小数も使えます。<br>セル参照は、列番号を表すアルファベットの大文字の後に行番号を表す数字を続けてセル位置を表します。<br>例） 2　数値2が計算に用いられます。<br>12.5　数値 12.5 が計算に用いられます。<br>A1　セル A1 に入っている数が計算に用いられます。 |
| 演算子 | ●式には、四則演算子、べき乗、モジュロ演算子、マイナス符号が使えます。<br>数値あるいはセル参照を演算子でつなぎ、数式どおりの式が入力できます。<br>例） 足し算　2+3　A1+B2<br>引き算　2-3　A1-B2<br>かけ算　2*3　A1*B2<br>割り算　2/3　A1/B2<br>これらを組み合わせていくらでも複雑な式が書けます。<br>例） A1+B2*100-C5 |
| 演算子の優先順位 | ●演算子の優先順位は次のように、大きな番号ほど高くなっています。<br>5　^　べき乗　2項演算<br>4　-　マイナス符号　単項演算<br>3　*　乗算　2項演算<br>3　/　除算　2項演算<br>2　%　モジュロ演算　2項演算<br>1　+　加算　2項演算<br>1　-　減算　2項演算 |
| ひとこと | ●%（モジュロ演算子）は簡単にいうと割り算の余りを返す演算子です。F-BASIC386のMOD 関数との違いは、返す値が負にはならないということです。つまり-5 % 3 は-1ではなく1となります。 |

6 TOWNSカルク

## 6. ビジネスの世界

**( ) カッコ**

● ( ) カッコを使って計算の優先順位を変えることができます。

- カッコはいくつでも繰り返し使えます。

  例) (A1+B2)*100-C5
  (A1+B2)*((100-C5)*20)

- 計算の優先順位は、一般的な計算式の規則と同じで、同じ演算記号が続けて書かれているときは計算は左からおこなわれます。
  ただしべき乗 ^ だけは、2 ^ 2 ^ 3 のように続けて書かれていると、右から、2 ^ (2 ^ 3)の優先順位で解釈されます。

**絶対番号指定（セル参照）**

● 式中で、セル位置を［列番号, 行番号］のように絶対番号で指定して参照できます。

- ブラケット［ の後に続けた数字で列番号を指定し、その後にカンマを入力して続けた数字で行番号を指定し、ブラケット ］で閉じます。
- 列番号の数字は、1がAに、2がB、3がCのように対応しています。

  例) [1,1]　　A1と同じ
  [1,1]+[2,2]*100-[3,5]　　A1+B2*100-C5 と同じ

**相対番号指定（セル参照）**

● セル参照で相対番号指定も使えます。

- 式の書き込みがおこなわれるセルからどれだけ離れているかでセル参照をします。

  例) [x-2,y+3]　　カレントセルは[x,y]で表され、この例はカレントセルから左に2つ、下に3つ離れたセルを示しています。

**セルポインタの移動**

● セルポインタを移動して相対番号指定のセル参照ができます。

- セルポインタ移動キーを押すと、白いセルポインタが現れます。セル参照をしたい位置へ移動して演算子を入力するか⏎キーを押すと、その位置の相対番号が入力行に入力されます。(⊟キーを押すと、その位置の絶対番号が入力行に入力されます。)

**ひとこと**

● 相対番号指定を使うと、セル参照の指定を、その式が書かれる位置によらない一般的な形で書くことができるので便利です。たとえば、繰越計算などでは、残高列のすべての行に同じ計算式を複写することで実行できます。

関数

●BASIC で使える関数はすべて式の中で使えます。

| | | |
|---|---|---|
| 平方根 | SQR(2) | ルート2を求める。 |
| 自然対数 | LOG(A1) | セル A1 に入っている数の自然対数を求める。 |
| 指数関数 | EXP(4+5) | 自然対数の底 e の4+5乗を求める。 |
| 最大整数 | INT(3.3) | 3.3 を越えない最大の整数、3 を求める。 |
| | INT(-3.3) | -3.3を越えない最大の整数、-4を求める。 |
| 切捨て | FIX(3.3) | 3.3 の小数点以下を切り捨てて、3 にする。 |
| | FIX(-3.3) | -3.3の小数点以下を切り捨てて、-3にする。 |
| 絶対値 | ABS(-5) | -5の絶対値、5 を求める。 |
| 符号 | SGN(A1-5) | セル A1 に入っている数から5を引いた数の符号を求める。プラスなら1、ゼロなら0、マイナスなら -1 が求まる。 |
| 乱数 | RND(1) | 乱数を求める。カッコの中の値は、BASIC の RND関数と同じ機能を持つ。 |
| サイン | SIN(A1) | セル A1 に入っている数のサインを求める。 |
| コサイン | COS(A1) | セル A1 に入っている数のコサインを求める。 |
| ﾀﾝｼﾞｪﾝﾄ | TAN(A1) | セル A1 に入っている数のタンジェントを求める。 |
| ｱｰｸﾀﾝｼﾞｪﾝﾄ | ATN(A1) | セル A1 に入っている数のアークタンジェントを求める。 |

これらを複合して、いくらでも複雑な式が書けます。

例） SQR( (A1-A2)^2 + (B1-B2)^2)

セルの中に入っている値を平面上の点の X 座標と Y 座標とみなして、(A1,B1)と(A2,B2)の間の距離を求める。

上記以外にBASIC にはない、SUM 関数が用意されています。

これは指定された範囲内のセルの総和を求める関数です。

| | |
|---|---|
| SUM(A1..A10) | セル A1 から A10までに入っている数の総和を求める。 |
| SUM(A1..C10) | セル A1 から A10、B1 から B10、C1 から C10 までに入っている数の総和を求める。 |
| SUM([1,1]..[4,5]) | セル [1,1] すなわち A1 から、セル [4,5] すなわち D5 までに入っている数の総和を求める。 |

# 6. ビジネスの世界

| | |
|---|---|
| ●データ入力以外の機能の使い方 | セルポインタを列番号表示、行番号表示および画面左上端へ移動するとコマンドメニューが表示され、いろいろな機能を使うことができます。 |
| 列を対象とした機能 | セルポインタを列番号表示へ移動すると、次のようなコマンドメニューが表示され、その列を対象として次のような機能が使えます。 |

```
D/小数部(2) C/ｺﾝﾏ*ON* R/折返し*ON* (H/左揃え J/中揃え K/右揃え)
T/固定表示*ON* P/列幅 (10)
```

| | |
|---|---|
| 小数部 | ●Dキーを押すと、小数点以下の表示桁数を、((列の桁数)-2)までの範囲で設定できます。 |
| コンマ | ●Cキーを押すと、3桁毎のコンマ表示をオン、オフできます。 |
| 折返し | ●Rキーを押すと、⏎キーが押されたときにセルポインタを折返す設定をオン、オフできます。<br>☞オンで列番号表示右端に[.]が表示されます。 |
| 左揃え | ●Hキーを押すと、文字データを左揃えにします。 |
| 中揃え | ●Jキーを押すと、文字データをセンター揃えにします。 |
| 右揃え | ●Kキーを押すと、文字データを右揃えにします。<br>☞左揃え、中揃え、右揃えはどれか選択されたものが有効となり、そのとき選択されているものが反転表示されます。 |
| 固定表示 | ●Tキーを押すと、表示固定列の設定をオン、オフできます。<br>☞表示固定列の設定をオンにすると、その列より左が緑色で表示され、セルポインタを表外へ移動しようとしてもスクロールされなくなります。<br>☞画面中央に表示されている列（最初に表示される表ではD列）まで固定できます。 |
| 列幅 | ●Pキーを押すと、表の大きさが最大 250桁を超えない範囲内で列の桁数を変更できます。変更には、多少時間がかかります。 |

**行を対象とした機能**
セルポインタを行番号表示へ移動すると、次のようなコマンドメニューが表示され、その行を対象として次のような機能が使えます。

```
T/固定表示 OFF
```

**固定表示**
● T キーを押すと、表示固定行の設定をオン、オフできます。
表示固定行の設定をオンにすると、その行より上が緑色で表示され、セルポインタを表外へ移動しようとしてもスクロールされなくなります。
10行目まで固定できます。

**表全体を対象とした機能**
セルポインタを表の左上端へ移動すると、次のようなコマンドメニューが表示され、表全体を対象として次のような機能が使えます。

```
W/表全体表示 L/表全体読込 A/自動計算*ON* Q/終了
```

**表全体保存**
● W キーを押すと、ドライブ名、ディレクトリ名、ファイル名を指定して、表全体をディスクに保存できます。
拡張子を入力しなくても、自動的に［.CAL］の拡張子が付けられます。
何も入力せずに ⏎ キーを押すと、保存の操作はキャンセルされます。

**表全体読込**
● L キーを押すと、ドライブ名、ディレクトリ名、ファイル名を指定して、ディスクに保存されている表を画面に取り込むことができます。
拡張子を入力しなくても、自動的に［.CAL］の拡張子が付けられます。
何も入力せずに ⏎ キーを押すと、読み込みの操作はキャンセルされます。

**自動再計算**
● A キーを押すと、自動再計算の機能をオン、オフできます。
オンからオフの操作では表内の計算式の参照を消去するために、オフからオンの操作では表内のすべての計算式を計算するために、多少時間がかかります。

**終了**
● Q キーを押すと、プログラムの実行を終了します。
プログラムを終了しても、PF10 キーを押すとTOWNS カルクの表に戻ることができます。

| | |
|---|---|
| 範囲指定時の機能 | [:]キーを押すと、その時点のセルポインタ位置から右下方向へ範囲の指定ができます。そして、指定された範囲について、次のような機能が使えます。 |
| 保存 | ●[W]キーを押すと、ドライブ名、ディレクトリ名、ファイル名を指定して指定された範囲のデータを保存できます。<br>拡張子を入力しなくても、自動的に拡張子.DATが付けられます。<br>何も入力せずに[↵]キーを押すと、保存の操作はキャンセルされます。<br>式のデータは、その計算結果が数値として保存されます。 |
| 読込 | ●[L]キーを押すと、ドライブ名、ディレクトリ名、ファイル名を指定して指定された範囲にディスクに保存されている表のデータを取り込むことができます。<br>指定されたファイルのデータは、最初から順番に取り込まれ、指定した範囲いっぱいになると切り捨てられます。<br>拡張子を入力しなくても、自動的に拡張子.DATが付けられます。<br>何も入力せずに[↵]キーを押すと、読み込みの操作はキャンセルされます。 |
| フィル | ●[F]キーを押すと、開始位置のセルのデータを指定された範囲内すべてのセルに複写します。 |
| 印刷 | ●[P]キーを押すと、指定された範囲内のデータをプリンタで印刷します。 |
| [ESC]キー | ●範囲指定をキャンセルします。 |
| [HOME]キー | ●範囲の設定中に、セルポインタを開始位置へ戻します。 |

**プログラムの概略**

TOWNS カルクのプログラムは、メインプログラムからサブルーチンを、サブルーチンからさらにサブルーチンを次々と呼びながらいろいろな処理を行っています。このため、サブルーチンの階層は次のように非常に複雑な構成になっています。

```
(MAIN)  ──┬─ 一番上の階層にあるメインプログラム
          ├→ DEF_TOKEN          コンパイラのトークンIDを定義
          ├→ DEF_INST           インストラクションコードを定義
          ├→ DEF_FUNCNAME       コンパイラが認識する関数名を定義
          ├→ DEF_MACRONAME      コンパイラが認識するマクロ(SUMのみ)を定義
          │
          ├→ INIT_EASY          簡単に使えるように表を初期化
          ├→ INIT_DISPVAR       表を管理する上での従属変数を正規化
          ├→ INIT_SHEET         表の表示上の部分を初期化
          └→ INIT_DISP          画面を初期化

EDIT    ──┬─ カーソルを動かし、表にデータを書く中で最も上の階層
          ├→ INPUT_CMD_GEN ──┬─ 表全体の処理をする
          │                  ├→ SAVE_ALL*            表全体を保存
          │                  ├→ LOAD_ALL*            表全体を読込
          │                  └→ SET_AUTOCALC*  自動再計算をON/OFF
          │
          ├→ INPUT_CMD_CLM ──┬─ 列についての処理をする
          │                  ├→ SET_FRAC             列の小数部を設定
          │                  ├→ SET_COMMA        列のコンマ表示を設定
          │                  ├→ SET_CARRIAGESTOP 列の折返しを設定
          │                  ├→ SET_STRLEFT          列を左揃えに設定
          │                  ├→ SET_STRCENTER        列を中揃えに設定
          │                  ├→ SET_STRRIGHT         列を右揃えに設定
          │                  ├→ SET_ROLCLM             SCROLL列を設定
          │                  └→ SET_CLMWDH                   列幅を設定
          │
          ├→ INPUT_CMD_ROW ──┬─ 行についての処理をする
          │                  └→ SET_ROLROW             SCROLL行を設定
          │
          ├→ CSR_UP*            カーソル上
          ├→ CSR_DOWN*          カーソル下
          ├→ CSR_RIGHT*         カーソル右
          ├→ CSR_LEFT*          カーソル左
          ├→ CSR_HOME*          カーソルをホームポジションへ
          ├→ CSR_RETURN*        カーソルリターン
          │
          ├→ INPUT_CMD_RNG ──┬─ 範囲指定をして処理をする
          │                  ├→ SPECIFY_RNG ──────┬─ 範囲指定をする
          │                  │                    ├→ CSR_DISP*
          │                  ├→ SAVE_RANGE        ├→ DISP*
          │                  │ (指定範囲を保存)   ├→ CSR_UP*
          │                  ├→ LOAD_RANGE        ├→ CSR_DOWN*
          │                  │ (指定範囲に読込)   ├→ CSR_RIGHT*
          │                  ├→ FILL              ├→ CSR_LEFT*
          │                  │ (指定範囲をフィル) ├→ CSR_HOME*
          │                  └→ PRINTOUT          └→ CSR_ERASE*
          │                    (指定範囲を印刷)
          │
          ├→ INPUT_NULL    ──┬─ スペースが押された時にセル消去
          │                  ├→ ERASE_CELL
          │                  └→ WRITE_CELL
          │
          ├→ INPUT_EXPR    ──┬─ 式の入力
          │                  ├→ INPUT_IN_STRING
          │                  ├→ COMPILE
          │                  ├→ ERASE_CELL
          │                  └→ WRITE_CELL
          ↓
```

```
                 |
                 |  ↓
                 |  |-> INPUT_NUM ────┬ 数字の入力
                 |  |                 |-> INPUT_IN_STRING
                 |  |                 |-> ERASE_CELL
                 |  |                 └-> WRITE_CELL
                 |  |
                 |  └-> INPUT_DATA ───┬ 文字の入力
                 |                    |-> INPUT_IN_STRING
                 |                    |-> ERASE_CELL
                 |                    └-> WRITE_CELL
ファイル入出力     |
    *SAVE_ALL    |  ──┬ 表全体を保存
                 |    |-> OPEN_FOR_SAVE ──┬ 表を保存するためにファイルを開く
                 |    |                   |-> FILE_WINDOW
                 |    |                   |-> INPUT_FILENAME
                 |    |                   └-> ANAL_FILENAME
                 |    |
                 |    |-> SAVE_ALL_SUB   表全体を保存する局所ルーチン
                 |    |-> CLOSE_FILE     ファイルを閉じる
                 |    └-> INIT_DISP      画面表示を初期化
                 |
   *SAVE_RANGE   |  ──┬ 指定範囲を保存
                 |    |-> OPEN_FOR_SAVE ──┬ 表を保存するためにファイルを開く
                 |    |                   |-> FILE_WINDOW
                 |    |                   |-> INPUT_FILENAME
                 |    |                   └-> ANAL_FILENAME
                 |    |-> SAVE_RANGE_SUB 範囲を保存する局所ルーチン
                 |    └-> CLOSE_FILE     ファイルを閉じる
                 |
    *LOAD_ALL    |  ──┬ 表全体に読込
                 |    |-> YESNO          キーボードから Y or N を受け付ける
                 |    |-> OPEN_FOR_LOAD ──┬ 表全体に読込むためにファイルを開く
                 |    |                   |-> FILE_WINDOW
                 |    |                   |-> INPUT_FILENAME
                 |    |                   └-> ANAL_FILENAME
                 |    |
                 |    |-> AUTOCALC_OFF   自動再計算を OFF
                 |    |-> LOAD_ALL_SUB   表全体に読み込む局所ルーチン
                 |    |-> CLOSE_FILE     ファイルを閉じる
                 |    └-> INIT_DISP      画面表示を初期化
                 |
   *LOAD_RANGE   |  ──┬ 指定範囲に読込
                 |    |-> OPEN_FOR_LOAD ──┬ 範囲に読込むためにファイルを開く
                 |    |                   |-> FILE_WINDOW
                 |    |                   |-> INPUT_FILENAME
                 |    |                   └-> ANAL_FILENAME
                 |    |
                 |    |-> AUTOCALC_OFF   自動再計算を OFF
                 |    |-> LOAD_RANGE_SUB 範囲に読み込む局所ルーチン
                 |    |-> CLOSE_FILE     ファイルを閉じる
                 |    └-> AUTOCALC_ON    自動再計算を ON
自動再計算        |
  *SET_AUTOCALC  |  ──┬ 自動再計算を ON / OFF
                 |    |-> AUTOCALC_OFF 自動再計算を OFF
                 |    └-> AUTOCALC_ON  自動再計算を ON
                 |
カーソル移動      |
    *CSR_UP      |  ──────> PT_UP
    *CSR_DOWN    |  ──────> PT_DOWN
    *CSR_RIGHT   |  ──────> PT_RIGHT
    *CSR_LEFFT   |  ──────> PT_LEFT
                 |
    *CSR_HOME    |  ───┬──> PT_LEFT      論理上のセルポインタを左に移動
                 |     |--> PT_RIGHT     論理上のセルポインタを右に移動
                 |     |--> PT_UP        論理上のセルポインタを上に移動
                 |     └--> PT_DOWN      論理上のセルポインタを下に移動
セル管理          |
    ERASE_CELL   |  ──────> UNLINK_REF  セルの参照関係を消す
```

```
WRITE_CELL      ─→ RECALC ─ 再計算をする
                            └→ ERA_ASSOC_CELLS
                            └→ CALC_ASSOC_CELLS ─→ UNLINK_REF
                                                 └→ CALC_EXPR
画面表示
     *DISP      ─┬ 表のイメージを画面に表示する
                 └→ DISP_ROW  表のイメージを1行だけ画面に表示する
 *CSR_DISP      ─┬ セルポインタを画面に表示する
                 ├→ CSR_CNT セルポインタ位置のセルの内容を画面に表示する
                 ├→ CSR_POS セルポインタの表示位置を計算する
                 └→ CSR_BOX セルポインタを画面に表示する
 *CSR_ERASE     セルポインタを消す
 *ADJ_IMAGE     ─┬ セルの内容に合わせて表のイメージを表全体について正規化する
                 └→ *ADJ_ROW ─ 表のイメージを1行だけ正規化する
                               ├→ ADJ_INDEX   表の縦横バー用の正規化
                               ├→ ADJ_STRING  文字用の正規化
                               └→ ADJ_NUMBER  数値用の正規化
  *ADJ_CLM      ─┬ 表のイメージを1列についてだけ正規化する
                 ├→ ADJ_INDEX   表の縦横バー用の正規化
                 ├→ ADJ_STRING  文字用の正規化
                 └→ ADJ_NUMBER  数値用の正規化
 *ADJ_CELL      ─┬ 表のイメージを一つのセルについてだけ正規化する
                 ├→ ADJ_INDEX   表の縦横バー用の正規化
                 ├→ ADJ_STRING  文字用の正規化
                 └→ ADJ_NUMBER  数値用の正規化
計算
 CALC_EXPR      ─┬ セルの式を計算する
                 └→ EXEC_MC ─ コンパイルされた式を計算
                              ├→ MISCEL ─ 他のセルを参照
                              │           └→ CALC_RECUR ─ 再帰的に計算
                              │                          └→ UNLINK_REF
                              ├→ UNARY   単項演算
                              ├→ BINARY  2項演算
                              ├→ LOAD    アキュームレーターにロード
                              ├→ STORE   アキュームレーターをストア
                              ├→ INC     スタック上のデータに1加算
                              ├→ JGT     大きいときジャンプ
                              ├→ POPN    スタックをN 戻す
                              └→ FUNC    関数を計算
INPUT_IN_STRING ─┬ 文字列を入力する
                 ├→ EDIT_IN_STRING ─ 文字列の中を編集する
                 │                   └→ EDIT_IN_STRING_SUB
                 ├→ CSR_UP*
                 ├→ CSR_DOWN*
                 ├→ CSR_RIGHT*
                 ├→ CSR_LEFT*
                 └→ CSR_HOME*
 UNLINK_ALL     ─┬ 全てのセルの参照関係を消す
                 └→ UNLINK_REF
 CALC_ALL_CELLS ─┬ 全てのセルを計算する
                 ├→ UNLINK_REF
                 └→ CALC_EXPR
```

```
式の評価
COMPILE
    EVAL_ADD      足算と引算を評価する
     EVAL_MOD     モジュロ演算を評価する
      EVAL_MUL    掛算と割算を評価する
       EVAL_UNA   マイナス符号を評価する
        EVAL_EXP  巾乗の演算を評価する
         EVAL_ELM  式の基本構成単位を評価する
          EVAL_CELLA  A1 の形のセル番号評価
          EVAL_MACRO  SUM を評価する
          EVAL_CELLN  [1,1]の形のセル番号評価
    REPORT_ERROR  コンパイル結果のエラーをレポートする

*MATCH_TOKEN  トークンが要求と一致するか調べ、次のトークンを得る
               GET_TOKEN ポインタを進めて次のトークンを得る

*OUT_CD     コンパイルしたコードを付け加える
*OUT_POPN   POPN用のコードを付け加える
*OUT_OPR1   1バイトのコードを付け加える
*OUT_OPR2   2バイトのコードを付け加える
*OUT_OPR8   8バイトのコードを付け加える
```

# 第2部

## F-BASIC386の操作のしかた

プログラムの作成はエディタで行います。エディタには、プログラムを作成し、実行するための機能があります。

エディタで作成したプログラムを、FMTOWNSが直接実行できる形式に変換するものが、コンパイラです。変換されたプログラムの実行速度はインタプリタで実行する場合にくらべると非常に早くなります。

ここでは、エディタの操作方法と、コンパイラの操作方法および操作上の注意点について解説します。



※コンパイラを使用するには「F-BASIC386コンパイラV2.1」が必要です。

# 第1章

## エディタの概要

この章では、エディタの画面を簡単に説明し、エディタ画面を使ったプログラムの作成・実行・保存・読み出しの流れを解説しています。

# 1. エディタの画面

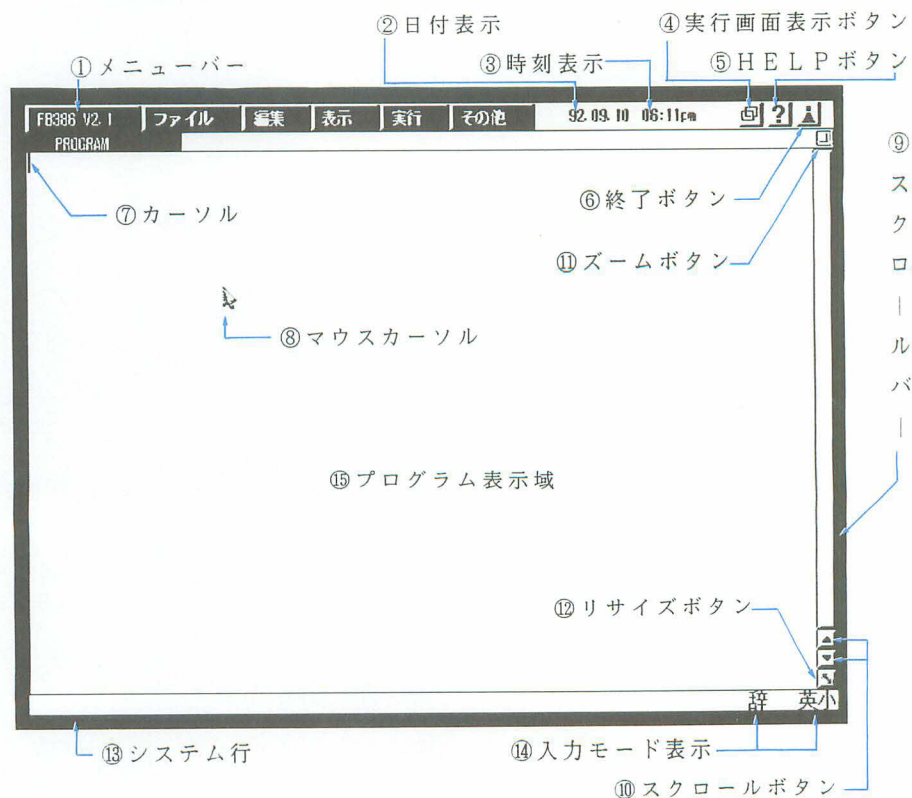
## エディタの画面

F-BASIC386は、プログラムを作るのに便利なプログラミング専用のエディタを持っています。F-BASIC386を起動すると、このエディタの画面が表示されます。エディタの画面では、プログラムを作ったり、サンプルプログラムを呼び出して編集したりするときに、マウス操作でいろいろな機能を使うことができます。

**ひとこと**

● F-BASIC386は、エディタの画面とは別にインタプリタの画面を持っています。今までのF-BASICでは、インタプリタの画面にエディタの機能を持たせていました。このため、プログラムを実行するとプログラムリストはいったん消去されました。でも、F-BASIC386ではプログラムを実行しても、どちらの画面も消されることなく残ります。ただし、256色モードを使って描いたグラフィック画面は、プログラムを終了すると消されます。

## 各部の名称

エディタの画面の各部の名称は、次のようになっています。

①メニューバー

●以下の６つのメニューがあります。

「FB386 V2.1」「ファイル」「編集」「表示」「実行」「その他」

これらを左クリックするか、対応するPFキー（「FB386 V2.1」を除いて左からそれぞれ順にPF1～PF5キーに対応している）を押すと、サブメニューが表示されます。（「サブメニュー」☞ 174ページ）

サブメニューの項目を左クリックすると、その機能が選択されます。ドラッグで選択することもできます。

ドラッグとは、メニューバーの項目上で、マウスの左ボタンを押し、そのままサブメニューの項目までマウスをずらしてから左ボタンを放すことを言います。

②日付表示

●システム内にあるカレンダーの日付を表示しています。

TownsMENU［設定］の中の［日付／時計］で、正しい日付に合わせることができます。

③時刻表示

●システム内にある時計の時刻を表示しています。

TownsMENU［設定］の中の［日付／時計］で、正しい時刻に合わせることができます。

④実行画面表示ボタン

●インタプリタの画面を表示させるためのボタンです。マウスカーソルをこのボタンに合わせて左クリックしている間だけ、インタプリタの画面が表示されます。

インタプリタの画面が256色モードになっていると表示できません。

⑤ＨＥＬＰボタン

●オンラインマニュアルを表示させるためのボタンです。マウスカーソルをこのボタンに合わせて左クリックすると、説明や構文を参照したり、印刷したりコピーしたりすることができます。（「ヘルプ機能の使い方」☞ 181ページ）

F-BASIC386Mを起動したときにだけ使えます。

⑥終了ボタン

●F-BASIC386を終了させるためのボタンです。マウスカーソルをこのボタンに合わせて左クリックすると、終了メッセージが表示されます。（「エディタの終了」☞ 180ページ）

| | |
|---|---|
| ⑦カーソル | ●プログラム表示域内にある赤い"｜"マークです。キーボードから入力した文字や記号は、このカーソル位置に常に挿入モードで書き込まれます。カーソルは、マウスカーソルを目的の位置に合わせて左クリックするか、→、←、↑、↓キーで移動させることができます。ただし、移動できる範囲は、プログラム表示域内に入力されているプログラム文の範囲内だけです。 |
| ⑧マウスカーソル | ●画面内に表示されている矢印です。マウスカーソルは、マウスを移動することで画面内を自由に動かすことができます。 |
| ⑨スクロールバー | ●プログラムの行数が多いため、全部を画面に表示できないとき、見たい場所を画面に表示するのに使います。<br>・スクロールバーの一番上の部分を左クリックすると、プログラムの先頭部分が表示されます。<br>・スクロールバーの中間の辺りを左クリックすると、プログラムの中間部分が表示されます。<br>・スクロールバーの一番下の部分を左クリックすると、プログラムの最後の部分が表示されます。 |
| ⑩スクロールボタン | ●プログラム表示域に表示されているプログラムを1行ずつ上下にスクロールするためのボタンです。マウスカーソルをこのスクロールボタンに合わせて左クリックすると、表示を矢印の方向へ1行（1論理行）ずつスクロールします。 |
| ⑪ズームボタン | ●プログラム表示域の大きさを、切り替えることができます。<br>このボタンを左クリックするごとに、最大の大きさ、またはリサイズボタンで設定した大きさになります。 |
| ⑫リサイズボタン | ●プログラム表示域の大きさを、変更することができます。<br>マウスカーソルをこのボタンに合わせてドラッグし、任意の大きさに設定します。 |
| ⑬システム行 | ●単語登録や漢字辞書機能を利用するとき、メッセージ表示、文字入力のときに使います。 |

| | |
|---|---|
| ⑭入力モード表示 | ●プログラム表示域に入力できる文字の種類やかな漢字変換機能の状態を表示します。<br>漢　：かな漢字変換モード。<br>全　：全角モード。表示のないときは半角モード。<br>英大／英小／かな／カナ：キーのシフト状態を表す。 |
| ⑮プログラム表示域 | ●プログラムを入力したり、編集したりする画面です。キーボードから入力された文字や記号は、カーソル位置に挿入されます。<br>タイトルバーにマウスカーソルを合わせてドラッグすると、プログラム表示域を画面内の任意の場所に移動できます。 |

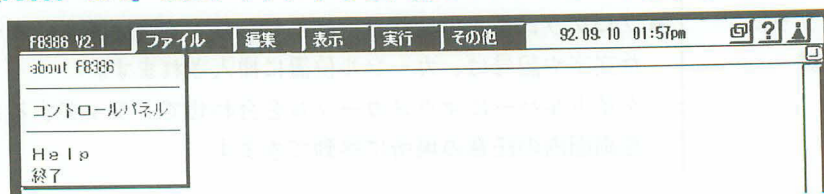
## 1 エディタの画面

**サブメニュー**

メニューバーの各メニューを左クリックすると、サブメニューが表示されます。

「FB386 V2.1」
「ファイル」☞ 175ページ
「編集」☞ 176ページ
「表示」☞ 177ページ
「実行」☞ 177ページ
「その他」☞ 178ページ

**● [FB386 V2.1] のサブメニュー**

[FB386 V2.1] を左クリックすると、次のサブメニューが表示されます。

それぞれの項目を左クリックすると、各機能が使えます。

**ご　注　意**

●サイドワーク機能が登録されている場合は、サイドワークメニューも表示されます。サイドワーク機能については、「Townsシステムソフトウェア」に添付のマニュアルを参照してください。

**about FB386**

●F-BASIC386のバージョン、レベルを表示します。

F-BASIC386V2.1L10

Copyright (c) FUJITSU LIMITED ,

FUJITSU PERSONAL COMPUTER SYSTEMS LIMITED 1992

確認

[確認] を左クリックすると、エディタの画面に戻ります。

**Help**

●オンラインマニュアルを表示します。（「ヘルプ機能の使い方」☞ 181ページ）

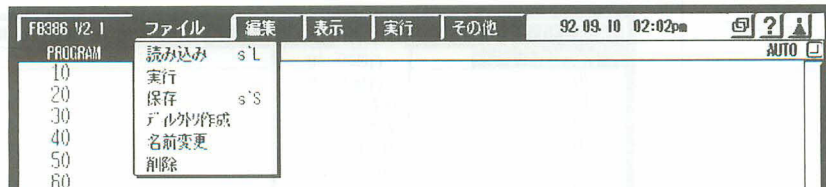
F-BASIC386Mを起動したときにだけ使えます。

**終了**

●F-BASIC386を終了します。（「エディタの終了」☞ 180ページ）

**●［ファイル］のサブメニュー**

［ファイル］を左クリックすると、次のようなサブメニューが表示されます。

FB386 V2.1　ファイル　編集　表示　実行　その他　92.09.10 02:02pm
PROGRAM　AUTO
10
20
30
40
50
60
読み込み　s^L
実行
保存　s^S
ディレクトリ作成
名前変更
削除

それぞれの項目を左クリックすると、各機能が使えます。

| | |
|---|---|
| 読み込み | ●保存してあるプログラムをエディタに読み込みます。（☞ 213ページ） |
| 実行 | ●保存してあるプログラムをエディタに読み込んで実行します。（☞ 206ページ） |
| 保存 | ●プログラムを保存します。（☞ 211ページ） |
| ディレクトリ作成 | ●サブディレクトリを作成します。（☞ 221ページ） |
| 名前変更 | ●ファイル名を変更します。（☞ 218ページ） |
| 削除 | ●ファイルを削除します。（☞ 219ページ） |

## 1 エディタの画面

### ● [編集] のサブメニュー

[編集] を左クリックすると、次のようなサブメニューが表示されます。

```
F8386 V2.1 | ファイル | 編集 | 表示 | 実行 | その他 | 92.09.21 10:16am
PROGRAM                                                      AUTO
30        一行UNDO  ^U
40        一行挿入  ^O
50        検索
60        置換
70        カット    s^X
80        コピー    s^C
90        ペースト  s^V
100       ポケットイン
110       ポケットアウト
```

それぞれの項目を左クリックすると、各機能が使えます。

**一行UNDO**
● カーソル位置のプログラム文の内容を、変更前の状態に復元します。(☞ 192ページ)

**一行挿入**
● プログラムの行と行の間を一行あけます。(☞ 193ページ)

**検索**
● 指定した文字列をプログラム文から探し出します。(☞ 193ページ)

**置換**
● 指定した文字列をプログラム文から探し出し、別の文字列に置き替えます。(☞ 195ページ)

**カット**
● エディタ上のプログラム文の一部を、範囲を指定して切り取り、専用のメモリに保存します。(☞ 197ページ)

**コピー**
● エディタ上のプログラム文の一部、またはヘルプ機能で表示した構文のウィンドウ内の文字列を、専用のメモリに複写して保存します。(☞ 197ページ)

**ペースト**
● 専用のメモリに保存されている内容を、カーソル位置に挿入します。(☞ 197ページ)

**ポケットイン**
● エディタ上のプログラム文をポケットに書き込みます。(☞ 198ページ)

**ポケットアウト**
● ポケットに書き込まれているテキストを、カーソル位置に取り出します。(☞ 199ページ)

● ［表示］の
サブメニュー

［表示］を左クリックすると、次のようなサブメニューが表示されます。

それぞれの項目を左クリックすると、各機能が使えます。

再表示 ●プログラムを正しい順序に表示し直します。（203ページ）

指定行表示 ●プログラムを、指定した行からエディタの画面に表示します。（203ページ）

Mark ●マークを行に設定し、その行にカーソルを飛ばします。（204ページ）

12/16dot font ●画面に表示する文字の大きさを変更します。（205ページ）

● ［実行］の
サブメニュー

［実行］を左クリックすると、次のようなサブメニューが表示されます。

それぞれの項目を左クリックすると、各機能が使えます。

Run ●エディタに表示しているプログラムを、最初から実行します。（207ページ）

Run 行番号 ●エディタに表示しているプログラムを、指定した行から実行します。（208ページ）

Run(Tron) ●エディタに表示しているプログラムを、トレースモードで実行します。（208ページ）

Continue ●中断したプログラムを再開します。（209ページ）

一行実行 ●行番号なしの命令文を一論理行だけ実行します。（208ページ）

● ［その他］のサブメニュー

［その他］を左クリックすると、次のようなサブメニューが表示されます。

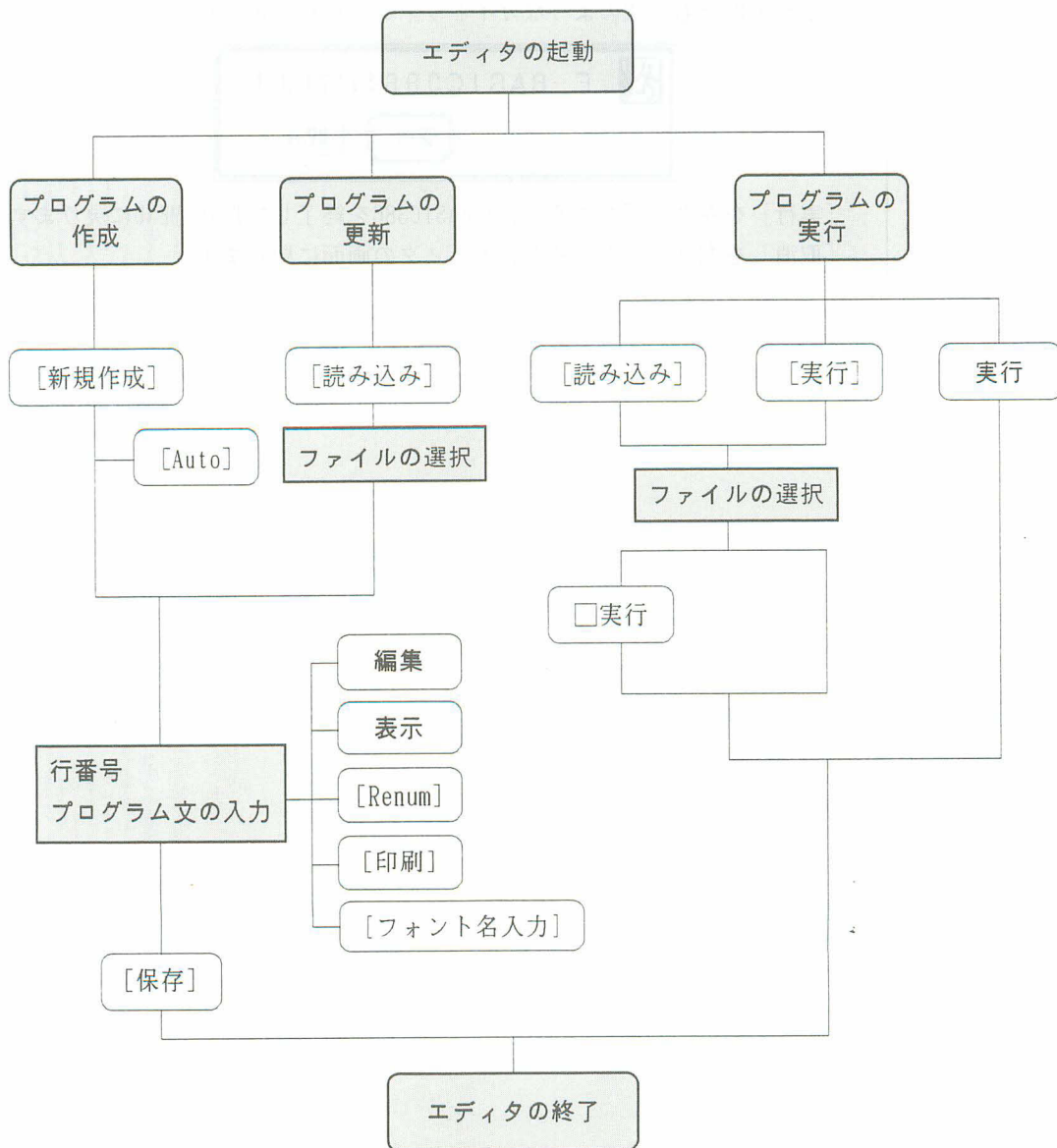
それぞれの項目を左クリックすると、各機能が使えます。

- **Auto** ●行番号を自動的に生成します。（☞ 186ページ）
- **Renum** ●行番号を等間隔に付け直します。（☞ 200ページ）
- **印刷** ●プログラムメモリ上のプログラムをプリンタに出力します。（☞ 201ページ）
- **フォント名入力** ●DEF FONT命令で指定するフォント名を選択します。（☞ 202ページ）
- **新規作成** ●メモリ上のプログラムをすべて消去します。（☞ 186ページ）

# 2. エディタの使い方

2

## プログラム作成の流れ

エディタでは、プログラムの作成、保存、読み出し、実行ができます。
ここでは、その簡単な流れを説明します。
☞各機能の詳しい内容については、第2章を参照してください。

### エディタの終了

F-BASIC386を終了するには、次の２つの方法があります。

- エディタの画面の終了ボタンを左クリックする。
- ［FB386 V2.1］のサブメニューで［終了］を選択する。

いずれの場合も、次のようなガイドウィンドウが表示されます。

［実行］を左クリックすると、F-BASIC386を終了してTownsMENUに戻ります。
［取消］を左クリックすると、エディタの画面に戻ります。

# 3. ヘルプ機能の使い方



**ヘルプ機能**

命令や関数の使い方やエディタの使い方がわからなくなったときに、画面に説明や構文を表示させ、参照したり、印刷したり、コピーしたりすることができます。特に、命令や関数の構文や例題は、直接プログラム文にコピーして使えます。

ヘルプ機能は［F-BASIC386M］でBASICを起動したときに、使用できます。

**ヘルプの使い方**

①エディタ画面右上の［HELP］ボタン、または［FB386 V2.1］のサブメニューで［Ｈｅｌｐ］を左クリックします。

ヘルプウィンドウが表示されます。

②構文を参照するのか、またはマニュアルを参照するのかを選択します。

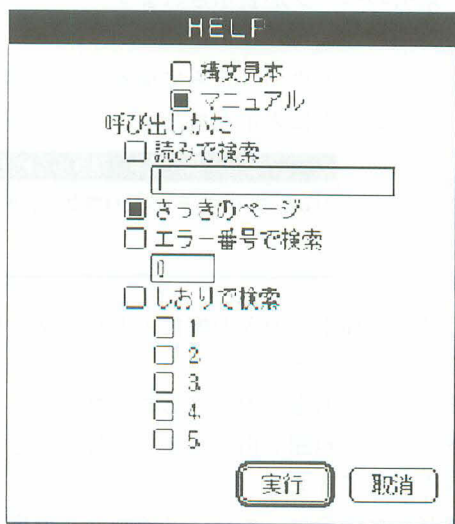
「構文見本」を選択し、「読みで検索」の入力欄に命令または関数を入力すると、該当する命令や関数の構文が表示されます。（☞ 182ページ）

「マニュアル」を選択すると、該当する命令や関数、用語の説明が、マニュアルの画面に表示されます。（☞ 183ページ）

③「マニュアル」を選択した場合は、呼び出しかたを指定します。

「読みで検索」「さっきのページ」「エラー番号で検索」「しおりで検索」の中から選択します。

④［実行］ボタンを左クリックすると、構文またはマニュアルが表示されます。

**ヘルプウィンドウ**

## 3　ヘルプ機能の使い方

**構文見本**　命令や関数の、構文だけを知りたいときに選択します。

**マニュアル**　命令や関数、用語の説明を見たいときに選択します。

**読みで検索**　入力欄に命令や関数、または用語を入力します。

- 入力できる文字は、ひらがな、カタカナ、英数記号です。
- 文字列を範囲指定してからヘルプ機能を呼び出すと、その文字列が表示されます。
- 命令や関数、用語を関連する言葉からさがせます。

**さっきのページ**　直前に表示したマニュアルのページを再度表示するときに選択します。

**エラー番号で検索**　エラー番号の内容を知りたいとき、入力欄にエラー番号を入力します。

**しおりで検索**　しおり登録されているマニュアルのページを見るときに選択します。
目的のしおりの名称を左クリックします。

**ご注意**

- 「読みで検索」の入力欄に何も入力しなかった場合は、マニュアルの目次のページが表示されます。
- 「読みで検索」の入力欄に入力された文字列に該当する命令や関数、用語が見つからなかった場合、「構文見本」を選択しているときはエラーメッセージが、「マニュアル」を選択しているときはその文字列に最も近い命令や関数、用語の索引のページが表示されます。

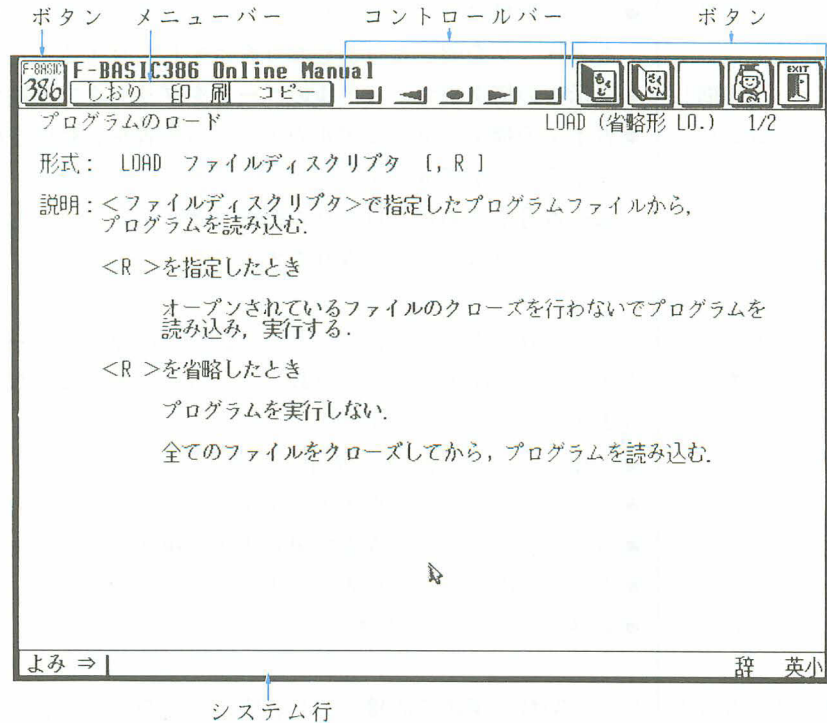
### 構文を参照する

ヘルプウィンドウの「読みで検索」の入力欄に入力した命令または関数の構文がガイドウィンドウに表示されます。

構文見本（複写する範囲を指定して下さい）
LOAD　ファイルディスクリプタ　[,R]
[コピー]　[終了]

- 「読みで検索」の入力欄に入力した文字列に該当する命令または関数が見つからなかったときは、エラーメッセージが表示されます。
  正しい文字列を入力し直してください。
- ドラッグで範囲を指定して［コピー］を左クリックすると、専用のメモリに内容を複写できます。
  複写した内容は、エディタの画面［編集］のサブメニュー［ペースト］でエディタに書き込むことができます。（☞ 197ページ）
- ［終了］を左クリックすると、ガイドウィンドウの表示を終了します。

## マニュアルを参照する

マニュアルの画面では、画面上部に表示されているメニューバーやコントロールバー、ボタンなどを、マウスで操作することでいろいろな機能が使えます。

### ●ボタンの機能

画面上部には、５つのボタンがあります。これらのボタンにマウスカーソルを合わせて左クリックすると、次のような機能が使えます。

●ヘルプ機能を終了し、エディタの画面に戻ります。

●目次を表示させて目的の命令や関数、用語の説明を探すことができます。

●ひらがな、あるいはアルファベットで頭文字を指定して索引を表示させ、目的の命令や関数、用語の説明や構文を表示させることができます。

●マニュアルの画面の操作方法を説明します。

●ヘルプ機能を終了し、エディタの終了メッセージを表示します。（☞ 180ページ）

## 3　ヘルプ機能の使い方

●メニューバーの機能

［しおり］［印刷］［コピー］のメニューバーにマウスカーソルを合わせて左クリックすると、サブメニューから次のような機能が使えます。

［しおり］

●表示中の構文やマニュアルのページにしおりをはさんだり、しおりをはさんであるページを開いたり、しおりをはずしたりすることができます。

［印刷］

●表示中の構文やマニュアルのページをプリンタで印刷することができます。

［コピー］

●表示中の構文やマニュアルのページの内容を、範囲を指定してエディタにコピーすることができます。

☞行番号を含めたプログラム例をエディタにコピーしたときは、エディタ上のプログラムにマージされます。

●コントロールバーの機能

ページ送りボタン（□、◁、○、▷、□）にマウスカーソルを合わせて左クリックすると、説明のページが複数（n）ページあるような場合、表示を次のように変更することができます。

| | |
|---|---|
| （左）□ | ●先頭のページ（1/n）を表示します。 |
| ◁ | ●１ページ前のページを表示します。 |
| ○ | ●直前に表示していた画面に戻ります。複数ページ前まで戻ることができます。 |
| ▷ | ●１ページ次のページを表示します。 |
| （右）□ | ●最終ページ（n/n）を表示します。 |

●システム行の機能

命令や関数、用語を直接入力し、実行キーを押すと、その命令や関数、用語の説明のページが表示されます。

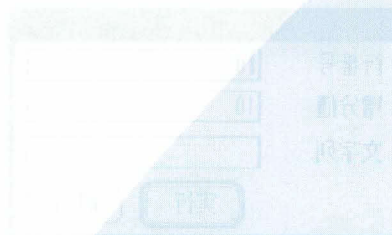
ひとこと

●説明内の青色の部分にマウスカーソルを合わせて左クリックすると、用語の説明やさらに詳しい説明のページにジャンプできます。

●詳しい操作の方法については、マニュアルの画面の「HELP」ボタンを左クリックして表示される説明、または「F-BASIC386 V2.1 リファレンス」を参照してください。

# 第2章

# エディタの操作

この章では、エディタでプログラムを作成したり実行したりするために必要な知識を解説しています。さらに、ディスクやフロッピィに保存してあるファイルの管理に便利な機能を解説します。

# 1. プログラムの作成

**プログラムの作成**

エディタの画面でプログラムを作成する場合は、次のような機能を使います。

- 新規作成
- Auto（行番号自動生成）

**●新規作成**

新たにプログラムを作成するときに、プログラムメモリをすべて消去して白紙の状態にします。

［その他］のサブメニューで［新規作成］を左クリックすると、プログラムメモリ上のプログラムがすべて消去されます。

**ご注意**

●この機能を実行すると、エディタ上のプログラムはすべて消去されます。必要なプログラムは、必ず保存しておいてください。

**ひとこと**

●［新規作成］を選択するかわりに、［一行実行］でNEW命令を実行しても、同じ結果になります。（「一行実行」☞ 208ページ）

**●Auto（行番号自動生成）**

新しい行を次々に入力していくとき、プログラム文の頭に付ける行番号を、改行するごとに自動的に生成します。

①［その他］のサブメニューで［Auto］を左クリックします。

次のようなガイドウィンドウが表示されます。

行番号：自動生成する最初の行番号
増分値：自動生成する行番号の間隔
文字列：行番号に続いて自動生成する文字列

②各項目を入力し、［実行］ボタンを左クリックするか、実行キーを押します。

プログラム表示域の右上に「AUTO」が表示され、AUTOモードになります。

1

ひとこと

- プログラム文を入力後、⏎キーを押すと、次の行の行番号が自動的に生成されます。
- 生成される行番号と同じ番号が既に存在する場合は、既存の行の命令の先頭にカーソルが移動します。
- すでに入力されている行番号の間隔より小さな値を増分値に設定すると、自動的に行番号を生成して行の挿入ができます。
- 指定した文字列が自動的に行番号のすぐ後に入力されるようになります。複数行の先頭に同じコマンドを入力するときに便利です。
- AUTOモードを解除するには、再度［その他］のサブメニューで［Auto］を左クリックするか、またはメニューバーの何かの項目を選択して実行します。

# 2.プログラムの編集

**プログラムの編集**

エディタの画面でプログラムを編集する場合は、次のような機能を使います。

エディタの基本機能（☞ 189ページ）

- カーソル移動
- 改行
- 文字の編集
- 行の編集

［編集］の機能（☞ 192ページ）

- 一行UNDO（アンドゥ）
- 一行挿入
- 検索
- 置換
- カット
- コピー
- ペースト
- ポケットイン
- ポケットアウト

［その他］の機能（☞ 200ページ）

- RENUM
- 印刷
- フォント名入力

［表示］の機能（☞ 203ページ）

- 再表示
- 指定行表示
- Mark
- 12/16dot font

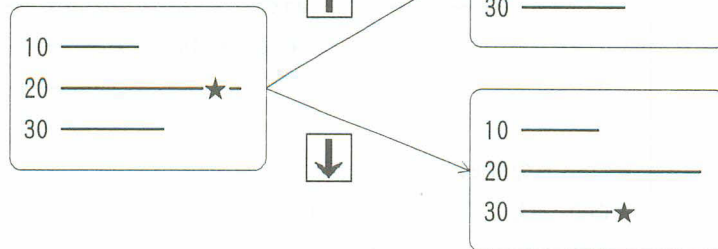
## エディタの基本機能

プログラムはキーボードのキーやマウスを使って、エディタの画面に作成します。プログラム文の入力や訂正は、すべてカーソル位置を基準に行います。

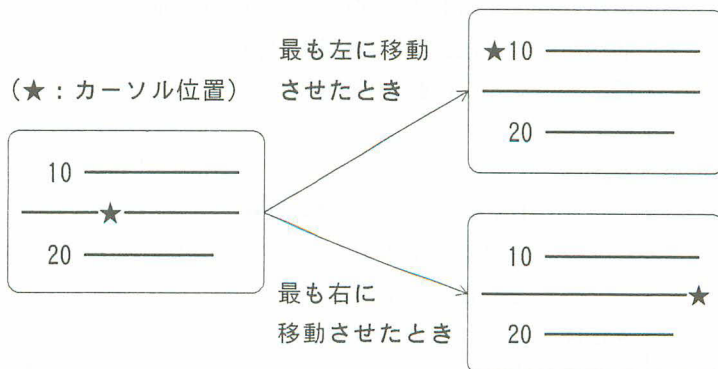
### カーソル移動

カーソル移動キーには、次のものがあります。

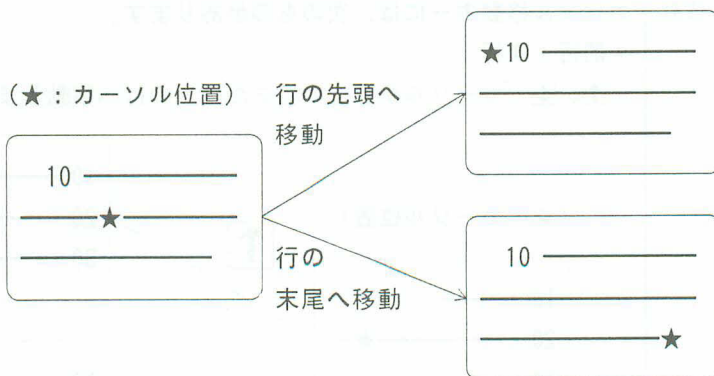
● 前行・次行

[↑]、[↓]：カーソルが、上の行または下の行へ移動します。

● 左右移動

[←]、[CTRL]＋[S]：カーソルが、1論理行の範囲内で左方向へ移動します。

[→]、[CTRL]＋[D]：カーソルが、1論理行の範囲内で右方向へ移動します。

## 2 プログラムの編集

● 行頭・行末

SHIFT＋←、SHIFT＋→：現在カーソルのある行（論理行）の先頭または末尾へ、カーソルが移動します。

● ワードの先頭

CTRL＋A：カーソルが、直前のワードの先頭文字位置へ移動します。

CTRL＋F：カーソルが、直後のワードの先頭文字位置へ移動します。

ワードとは、空白、コンマ（, ）、カッコなどで区切られた文字列をいいます。

**改行** 改行を行うときは、⏎キーを押します。

現在カーソルのある論理行の次の行の先頭に、カーソルが移動します。

AUTOモードでは、次の行に行番号を生成し、その行番号の右、命令を入力する位置にカーソルが移動します。

**文字の編集**

● 削除 1

後退（JISキーボードの場合は←）

：後退キーを押すと、カーソルの左側の 1 文字を削除し、それ以降の文字を左詰めします。

● 削除 2

削除：カーソルの右側の 1 文字を削除し、それ以降の文字を左詰めします。

● 削除 3

SHIFT＋削除：カーソルの位置からその行の終わりまでを削除します。

● 挿入

キーボードからの入力は、いつも挿入モードです。

行の編集

● 行の挿入

現在の行番号の中間の新しい行番号で、あらたなプログラム文を入力します。入力する位置はどこでもかまいません。

入力後、「再表示」で挿入されたことを確認できます。（「再表示」☞ 203ページ参照）

☝ある行のすぐ下に入力したいときは、「編集」のサブメニューから「一行挿入」（☞ 193ページ）を選択します。

● 行の複写

画面上に表示されている行番号を書き替えると、その行が書き替えた行番号に複写されます。

入力後、「再表示」で複写されたことを確認できます。

☝行番号だけでなく、命令文も書き替えると、書き替えた命令文が新しい行として追加されます。

☝行の中の範囲を指定して複写したいときに、「コピー」（☞ 197ページ）で範囲を指定して専用のメモリに保存し、「ペースト」（☞ 197ページ）で別の場所へ複写します。

● 行の削除

画面上に表示されている行の行番号だけを残して命令文を全部削除するか、または、削除したい行番号だけを入力します。（入力する位置は、どこでもかまいません）

入力後、「再表示」で削除されたことを確認できます。

☝連続した多くの行を削除したいときには、「一行実行」（☞ 208ページ）でDELETE命令を実行する方が便利です。

## 2 プログラムの編集

[編集] の機能

プログラム文を誤って変更したときに元の状態に復元したり、行と行の間をあけたり、文字列を探したり、専用のメモリやポケットを使って文字列を移動したり複写したりします。

一行UNDO（アンドゥ）

**カーソル位置のプログラム文の内容を、変更前の状態に復元します。**

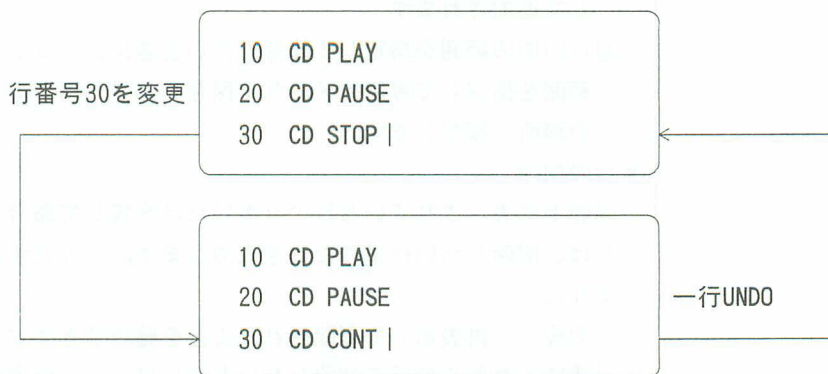
プログラムを間違って変更または削除したとき、カーソルを別の論理行に移動する前ならば内容を復元できます。

● **[編集] のサブメニューで [一行UNDO] を左クリックします。**

CTRL＋Uキーを押しても、同じ結果になります。

↵キーを押す前のカーソル行の内容が復元されます。

行番号を書き換えた行は、元の行番号に戻ります。（行の複写が取り消されます。）

（|：カーソル）

ご注意

●すでにカーソルを別の論理行へ移動してしまったときは、復元できません。

一行挿入

**プログラムの行と行の間に一行空けることができます。**

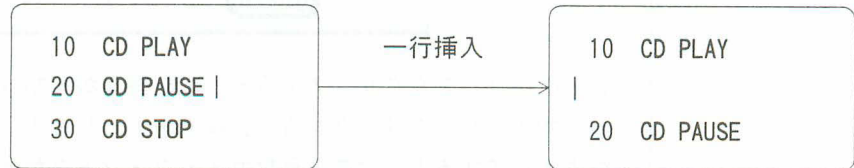
プログラムの途中に一行新しい行を挿入するのに、前後の行を参照しながら入力したいような場合、その位置に一行空けて入力することができます。

[一行実行]（☞ 208ページ）をエディタ上で行いたいときにも利用できます。

● **[編集]のサブメニューで[一行挿入]を左クリックします。**

CTRL＋Qキーまたは SHIFT＋挿入キーを押しても、同じ結果になります。

カーソルのある行と前の行との間を一行空け、空いた行の先頭にカーソルが移動します。　（|：カーソル）

検索

**指定した文字列を、プログラム文から探し出します。デバッグのときなどに利用できます。**

① **[編集]のサブメニューで[検索]を左クリックします。**

[検索]を選択する前に文字列をドラッグで範囲指定しておくと、その文字列が「検索文字列」のところに表示されます。

次のようなガイドウィンドウが表示されます。

検索文字列　：検索する文字列
英小文字区別：英大文字と小文字を区別するか否か
↑方向検索　：カーソル位置から上方向に検索
↓方向検索　：カーソル位置から下方向に検索

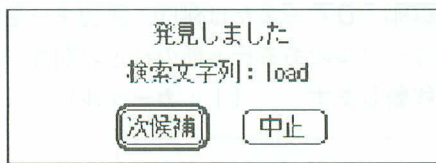
② **「検索文字列」を入力し、「英小文字区別」を設定します。**

③［↑方向検索］または［↓方向検索］を左クリックします。

［取消］を左クリックすると、何もしないでエディタの画面に戻ります。

検索中は、検索文字列を表示したガイドウィンドウが表示されます。

④該当する文字列が見つかると、その文字列を反転表示し、次のようなガイドウィンドウが表示されます。

発見しました
検索文字列：load
次候補　中止

［次候補］を左クリックすると、再び検索が開始されます。

［中止］を左クリックすると、検索を中止します。

⑤検索が終了すると、次のようなガイドウィンドウが表示されます。

見つかりません
確認

［確認］を左クリックすると、検索を終了し、エディタの画面に戻ります。

●キーボードからの操作

「検索」は、キーボードからも操作できます。

CTRL＋L：カーソル位置から1ワード（検索する文字列）を取り込みます。

CTRL＋N：↓方向検索。文字列を探し出して、その位置へカーソルを移します。

CTRL＋P：↑方向検索。文字列を探し出して、その位置へカーソルを移します。

置換

指定した文字列を、プログラム文から探し出し、別の文字列に置き替えます。変数名やラベル名を変更するときに利用できます。

①［編集］のサブメニューで［置換］を左クリックします。

次のようなガイドウィンドウが表示されます。

文字列置換
置換対象文字列
置換文字列
□英小文字区別
■カーソル以下一部
□カーソル以下全部
置換　終了

置換対象文字列　：置き替えられる文字列
置換文字列　　　：置き替える新文字列
英小文字区別　　：英大文字と小文字を区別するか否か
カーソル以下一部：文字列を発見するごとに、反転表示して確認する場合
カーソル以下全部：全部の文字列を一括して置き替える場合

②「置換対象文字列」「置換文字列」を入力し、各項目を設定します。

③［置換］を左クリックします。

［終了］を左クリックすると、何もしないでエディタの画面に戻ります。

検索中は、置換対象文字列と置換文字列がウィンドウに表示されます。

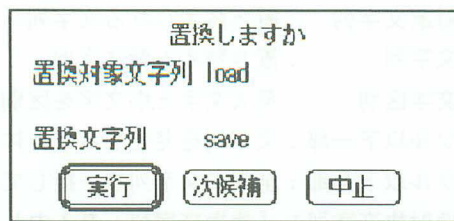
「カーソル以下全部」に設定した場合、ESC キーを押すと処理が中断されます。

「カーソル以下全部」に設定したとき

④自動的に置換を行い、終了すると置換した文字列の個数を表示します。

[確認] を左クリックすると、置換を終了し、エディタの画面に戻ります。

「カーソル以下一部」に設定したとき

④該当する文字列が見つかると、その文字列を反転表示し、次のようなガイドウィンドウが表示されます。

[実行] を左クリックすると、文字列が置換され、再び検索が開始されます。

[次候補] を左クリックすると、文字列は置換されず、再び検索が開始されます。

[中止] を左クリックすると、置換を中止します。

⑤置換が終了すると、置換した文字列の個数を表示します。

[確認] を左クリックすると、置換を終了し、エディタの画面に戻ります。

カット

エディタ上のプログラム文の一部を、専用のメモリに保存します。

①カットしたい部分の先頭でマウスの左ボタンを押し、ドラッグで範囲を指定します。

範囲が1ワードのときは、そのワードの任意の箇所をダブルクリックして指定することもできます。

②［編集］のサブメニューで［カット］を左クリックします。

SHIFT＋CTRL＋Xキーを押しても、同じ結果になります。

指定した範囲の文字列がエディタから削除され、メモリに保存されます。

ひとこと

● メモリに保存した内容は、［ペースト］で他の場所へ貼り込むことができます。

● 間違ってカットした文字列を元に戻したいときは、［一行UNDO］（192ページ）を行うか、または同じ位置で［ペースト］を行います。

コピー

エディタ上のプログラム文の一部、またはヘルプ機能で表示した構文のガイドウィンドウ内の文字列を、専用のメモリに複写して保存します。

①コピーしたい部分の先頭でマウスの左ボタンを押し、ドラッグで範囲を指定します。

範囲が1ワードのときは、そのワードの任意の箇所をダブルクリックして指定することもできます。

②［編集］のサブメニューで［コピー］を左クリックします。

SHIFT＋CTRL＋Cキーを押しても、同じ結果になります。

指定した範囲の文字列が専用のメモリに保存されます。文字列はエディタから削除されません。

ひとこと

● メモリに保存した内容は、［ペースト］で他の場所へ貼り込むことができます。

ペースト

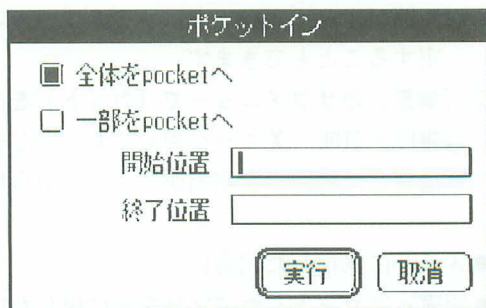
専用のメモリに保存されている内容を、カーソル位置に挿入します。

①文字列を挿入する位置にカーソルを移動します。

②［編集］のサブメニューで［ペースト］を左クリックします。

SHIFT＋CTRL＋Vキーを押しても、同じ結果になります。

メモリの内容が、カーソル位置に挿入されます。

ひとこと

● メモリの内容は、新たに［カット］または［コピー］を行うまで、同じものが残っています。同じ内容を何度でも挿入できます。

**ポケットイン**

プログラムの指定した範囲をポケットに書き込みます。

① ［編集］のサブメニューで［ポケットイン］を左クリックします。

次のようなガイドウィンドウが表示されます。

ポケットイン
■ 全体をpocketへ
□ 一部をpocketへ
開始位置
終了位置
実行　取消

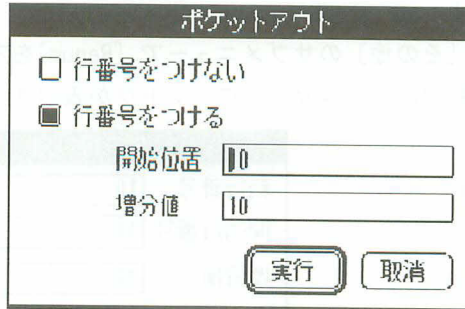
全体をポケットへ：表示中のプログラムの全部を書き込む
一部をポケットへ：表示中のプログラムの一部を書き込む
開始位置：書き込みを開始する位置
空白にすると、プログラムの先頭から書き込まれます
終了位置：書き込みを終了する位置
空白にすると、プログラムの末尾まで書き込まれます

②全体か一部かを選択します。一部を選択した場合は、開始位置と終了位置を指定します。

③［実行］ボタンを左クリックするか、実行キーを押します。

［取消］を左クリックすると、書き込まれず、エディタの画面に戻ります。

プログラムの指定した範囲がポケットに書き込まれます。

**ご注意**

- 開始位置または終了位置を入力するときは、行番号またはラベル名のどちらかに統一して指定します。両方を混在させて指定することはできません。
- ポケットに書き込めるプログラムの容量は、４ＫＢです。F-BASIC386をハードディスクから起動した場合は、ポケットの容量を超えた分のプログラムは一時的にハードディスクに書き込まれます。
- ポケットに書き込めるデータは１つだけです。［ポケットイン］を行うと、前のデータは消去されます。
- ポケットがTownsOS にインストールされていない場合は、この処理は選択できません。
- メモリ上にプログラムがない場合は、この処理は選択できません。

ポケットアウト

ポケットに書き込まれているテキストを、エディタに表示中のプログラムにマージします。

①［編集］のサブメニューで［ポケットアウト］を左クリックします。

次のようなガイドウィンドウが表示されます。

行番号をつけない：テキストに行番号をつけずにマージする場合
行番号をつける　：テキストに行番号をつけながらマージする場合
　　　開始位置：テキストにつける最初の行番号
　　　増分値　：テキストにつける行番号の間隔

②各項目を設定します。

③［実行］を左クリックするか、実行キーを押します。

ポケットに書き込まれているテキストがマージされます。

ご注意

- ポケットに書き込まれているデータがテキスト以外の場合はエラーになります。
- 開始位置、増分値を省略すると、10を指定したとみなされます。
- ポケットがTownsOS にインストールされていない場合は、この処理は選択できません。

**［その他］の機能**

行番号を付け替えたり、作成したプログラムを印刷したり、DEF FONT命令で使うフォントの種類を選択したりします。

**Renum（行番号の付け替え）**

プログラムの編集後、行の追加や複写などで乱れた行番号を、等間隔に付け直します。

① ［その他］のサブメニューで［Renum］を左クリックします。

次のようなガイドウィンドウが表示されます。

新行番号　：付け直す行番号の最初の番号
　　　　　　省略すると、10を指定したとみなされます
開始行番号：行番号を付け直す行
　　　　　　省略するか、空白を入力すると、プログラムの先頭を指定したとみなされます
増分値　　：付け直す行番号の間隔

②各項目を入力し、［実行］を左クリックするか、実行キーを押します。

指定した開始行番号以降が、新しい増分値で新行番号に付け替えられます。

印刷

プログラムメモリ上のプログラムの、全部または一部をプリンタに出力します。

①［その他］のサブメニューで［印刷］を左クリックします。

次のようなガイドウィンドウが表示されます。

■ 全体印刷
□ 部分印刷
開始位置
終了位置
実行　取消

全体印刷：表示中のプログラムの全部を印刷する場合

部分印刷：表示中のプログラムの一部を印刷する場合

　開始位置：印刷を開始する位置

　　　　　　空白にすると、プログラムの先頭から印刷されます

　終了位置：印刷を終了する位置

　　　　　　空白にすると、プログラムの末尾まで印刷されます

②全体印刷か部分印刷かを選択します。部分印刷を選択した場合は、開始位置と終了位置を入力します。

③［実行］ボタンを左クリックするか、実行キーを押します。

［取消］を左クリックすると、印刷は行われず、エディタの画面に戻ります。

プログラムの指定した範囲が印刷されます。

ご注意

- 開始位置および終了位置を入力するときは、行番号とラベル名のどちらかに統一して指定します。両方を混在させて指定することはできません。
- プリンタの準備ができていないと、エラーメッセージが表示されます。プリンタを印刷できる状態にして、再度指定し直してください。

ひとこと

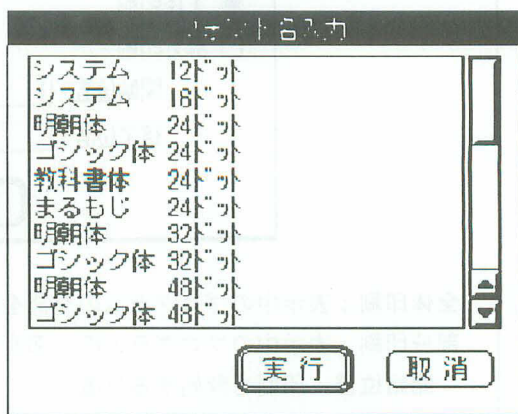
- 印刷を途中で終わらせる場合は、BREAK キーを押します。
- ［印刷］を選択するかわりに、［一行実行］（☞ 208ページ）でLLIST命令を実行しても、印刷できます。この場合、全体印刷になります。

フォント名入力

DEF FONT命令で指定するフォント名を一覧表示し、そのフォント名をカーソル位置に挿入します。

①フォント名を挿入する位置にカーソルを移動します。

②［その他］のサブメニューで［フォント名入力］を左クリックします。

次のようなガイドウィンドウが表示されます。

③目的のフォント名を選択し、［実行］を左クリックすると、カーソル位置にそのフォント名が挿入されます。

［取消］を左クリックすると、何も行われずにガイドウィンドウが消え、エディタの画面に戻ります。

**[表示] の機能**

プログラムの編集中に、行の追加や複写で乱れた画面を表示し直したり、指定した行からプログラムを表示したり、指定した行へカーソルを飛ばしたり、画面に表示する文字の大きさを変更したりすることができます。

**再表示**

プログラムを正しい順序で表示し直します。

① [表示] のサブメニューで [再表示] を左クリックします。

[再表示] を左クリックするかわりに、CTRL＋Kまたは HOME キーを押しても、同じ結果になります。

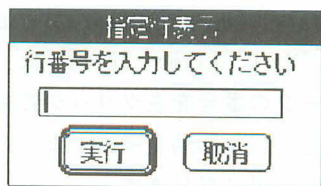
[再表示] を行うと、次のようになります。

- 行番号が順番になっていなかった行は、正しい順序で表示されます。
- 行番号は 5 桁に揃えられます。
- プログラム文を修正した場合は、修正後の正しい状態が表示されます。
- 省略形で入力した命令は、完全なつづりで表示されます。
- 255文字以上入力した行は、有効な文字のみ表示されます。

**指定行表示**

プログラムを、指定した行からエディタの画面に表示します。

① [表示] のサブメニューで [指定行表示] を左クリックします。

次のようなガイドウィンドウが表示されます。

②行番号またはラベル名を入力し、[実行] ボタンを左クリックするか、実行キーを押します。

その行から、プログラムがエディタの画面に表示されます。

スクロールバーの適当な位置を左クリックすることにより、画面にプログラムの見たい所を表示させることもできます。(☞ 172ページ)

Mark

[Mark] では、次の3つの機能が使えます。

Mark：マークを設定します。

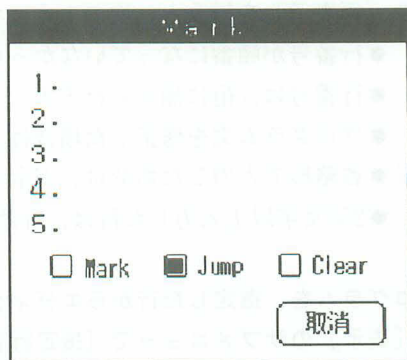
Jump：マークしてある行にカーソルを飛ばします。

Clear：設定してあるマークを解除します。

①マークの設定をする場合は、マークしたい行にカーソルを移動します。

カーソルの位置は、マークしたい行内なら、どこでもかまいません。

② [表示] のサブメニューで、[Mark] を左クリックします。

次のようなウィンドウが表示されます。

③ [Mark] [Jump] [Clear]から処理を左クリックで選択します。

[Mark] [Jump] [Clear]は、それぞれM、J、Cのキーでも選択できます。

④マークの番号を左クリックで指定します。

カーソル移動キー（↑↓）でマークの番号を選択し、↵キーを押しても、同じ結果になります。

②で選択した処理により、それぞれ次のような結果になります。

- [Mark] を選択したときは、カーソルのある行にマークが設定され、その番号に行番号と行の内容が登録されます。
- [Jump] を選択したときは、その番号に設定されている行にカーソルが移動します。
- [Clear] を選択したときは、その番号に設定されているマークが解除されます。

**ひとこと**

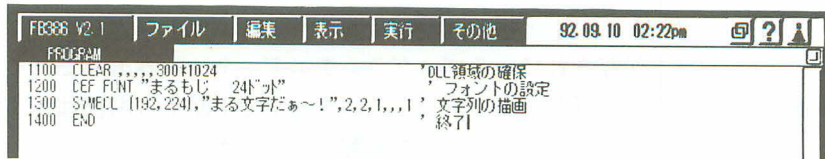
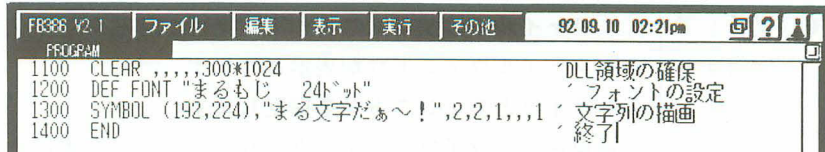
●次の操作を行うと、設定してあるマークはすべて解除されます。
- ［ファイル］のサブメニューで［実行］または［読み込み］を行う
- ［その他］のサブメニューで［新規作成］を実行する

●マークの設定をする場合は、①の操作のあと、CTRL＋実行キーを押してもマークが設定できます。

**12/16dot font**

**エディタに表示中の文字の大きさ（ドット）を切り替えます。**

●［表示］のサブメニューの［12/16dot font］を左クリックするごとに、12dotと16dotが切り替わります。

〈12dot〉

〈16dot〉

# 3. プログラムの実行

**プログラムの実行**

プログラムを実行します。実行には、次のような方法があります。

保存してあるファイルの実行

- 「ファイル」のサブメニューから「実行」を選択する
- 「ファイル」のサブメニューから「読み込み」を選択し、「□実行」に設定する

エディタに表示中のプログラムの実行

- 全体実行…………Run
- 部分実行…………Run 行番号
- トレースオン……Run(Tron)
- 一行実行

**保存してあるファイルの実行**

保存されているプログラムをエディタに読み出して、インタプリタで実行します。

実行しているプログラムを途中で中断する場合は、「プログラム実行の中断」（☞ 209ページ）を参照してください。

プログラムの実行が終了したときのメッセージについては、「プログラム実行の終了」（☞ 210ページ）を参照してください。

**●「実行」で実行する**

①「ファイル」のサブメニューで「実行」を左クリックします。

ファイルウィンドウが表示されます。

②ドライブとディレクトリを指定します。

現在のドライブ／ディレクトリのファイルを実行する場合は、ドライブ／ディレクトリを移動する必要はありません。

ドライブ／ディレクトリの変更のしかたは、「ファイルウィンドウの操作」（☞ 217ページ）を参照してください。

③ファイル名リストの中から、実行するファイルを指定します。

④［実行］を左クリックします。

指定したファイルが読み込まれ、実行されます。

**ご注意**

- 実行の前にエディタに表示されていたプログラムは、実行終了後、実行したプログラムがエディタに表示されるため、消去されます。必要なプログラムは必ず保存してから実行してください。

● 「読み込み」で実行する

①「ファイル」のサブメニューで「読み込み」を左クリックします。

ファイルウィンドウが表示されます。

②「□実行」を左クリックします。

③ドライブとディレクトリを指定します。

現在のドライブ／ディレクトリのファイルを実行する場合は、ドライブ／ディレクトリを移動する必要はありません。

ドライブ／ディレクトリの変更のしかたは、「ファイルウィンドウの操作」（☞ 217ページ）を参照してください。

④ファイル名リストの中から、実行するファイルを指定します。

⑤［実行］を左クリックします。

指定したファイルが読み込まれ、実行されます。

ご注意

● 実行の前にエディタに表示されていたプログラムは、実行終了後、実行したプログラムがエディタに表示されるため、消去されます。必要なプログラムは必ず保存してから実行してください。

## メモリ上のプログラムの実行

エディタの画面に表示されているプログラムを、インタプリタで実行します。

実行しているプログラムを途中で中断する場合は、「プログラム実行の中断」（☞ 209ページ）を参照してください。

プログラムの実行が終了したときのメッセージについては、「プログラム実行の終了」（☞ 210ページ）を参照してください。

Run（全体実行）

プログラムを最初から実行します。

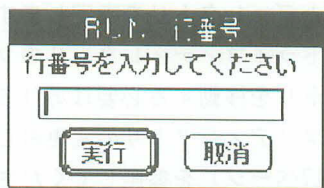
● ［実行］のサブメニューで［Run］を左クリックします。

## 3 プログラムの実行

**Run 行番号（部分実行）**

プログラムを途中の行から実行します。

①［実行］のサブメニューで［Run 行番号］を左クリックします。

次のようなガイドウィンドウが表示されます。

②行番号またはラベル名を入力し、［実行］を左クリックするか実行キーを押します。

指定した行からプログラムが実行されます。

**Run(Tron)（トレースオン）**

実行された行の行番号をテキスト画面に表示しながら、プログラムを最初から実行します。デバッグのときに利用します。

● ［実行］のサブメニューで［Run(Tron)］を左クリックします。

**一行実行**

エディタ上のプログラムとは関係なく、行番号なしの命令文を1論理行だけ実行します。

一行実行には、次のような実行方法があります。

<u>［実行］のサブメニューで実行する</u>

①［実行］のサブメニューで［一行実行］を左クリックします。

次のようなガイドウィンドウが表示されます。

②キーボードから命令文を入力し、［実行］を左クリックするか、実行キーを押します。

命令が実行されます。

<u>エディタで直接実行する</u>

●エディタの編集画面で、行番号なしに命令を入力し、⏎キーを押すと、すぐにその命令が実行されます。

## プログラム実行の中断

実行中のプログラムを、強制的に中断することができます。

プログラムを中断するには、次のような方法があります。

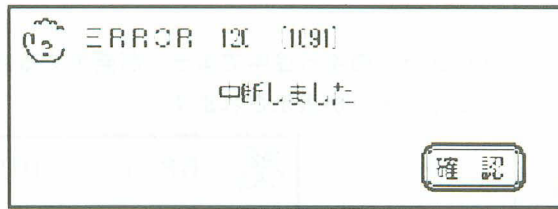
**●命令による中断**

- STOP命令を実行する。
- ON ERROR GOTO命令がなくて、ERROR命令を実行する。

**●キーによる中断**

- BREAK キーを押す。
- INPUT命令によってデータの入力状態にあるときに、CTRL＋Cキーを押す。

STOP OFF命令で、これらのキーによる中断を無効にすることができます。

プログラムが中断されると、エラー音が鳴り、次のようなガイドウィンドウが表示されます。

［確認］を左クリックすると、エディタの画面に戻ります。

**ご注意**

●ファイルのオープン中などに中断したときは、そのままの状態でフロッピィディスクを差しかえたりすると、ファイルが破壊されることがあります。
フロッピィディスクを差しかえる前に、一行実行でEND命令を実行してください。

## 中断したプログラムの再開

中断したプログラムを、中断した位置から再開させることができます。

● ［実行］のサブメニューで［Continue］を左クリックします。

INPUT命令で中断したときは、そのINPUT命令から、その他のときは、中断した命令の次の命令から再開されます。

**ご注意**

●中断後にプログラムを編集すると、再開できなくなります。「続行できません」というメッセージが表示されます。

●MOUSE命令、SPRITE命令などの命令で中断したときも、再開されません。

## 3 プログラムの実行

**プログラム実行の終了**

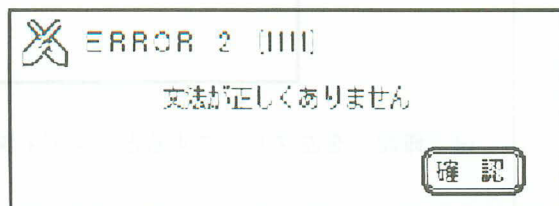
プログラムを実行すると、結果に応じて次のようなガイドウィンドウが表示されます。

●正常終了

プログラムの実行が問題なく終わったときは、次のようなガイドウィンドウが表示されます。

[確認] を左クリックすると、エディタの画面に戻ります。

●エラー

プログラムの実行途中でエラーが発生すると、エラー音が鳴り、次のようなガイドウィンドウが表示されます。

[確認] を左クリックすると、エディタの画面に戻り、エラーが発生した行にカーソルが移動します。

**ひとこと**

●エラー番号とエラー内容については、「F-BASIC386 V2.1 リファレンス」の「エラーメッセージ一覧」を参照してください。

●エラーのガイドウィンドウが表示されたすぐ後に [HELP] を呼び出すと、ヘルプウィンドウの [エラー番号で検索] の欄にそのエラー番号が表示され、エラーの内容を確認することができます。（「ヘルプ機能の使い方」☞ 181ページ）

# 4. プログラムの保存

**プログラムの保存**

エディタ上のプログラムにファイル名を付け、ドライブを指定して保存します。一行実行でSAVE命令を実行しても、同じ結果になります。

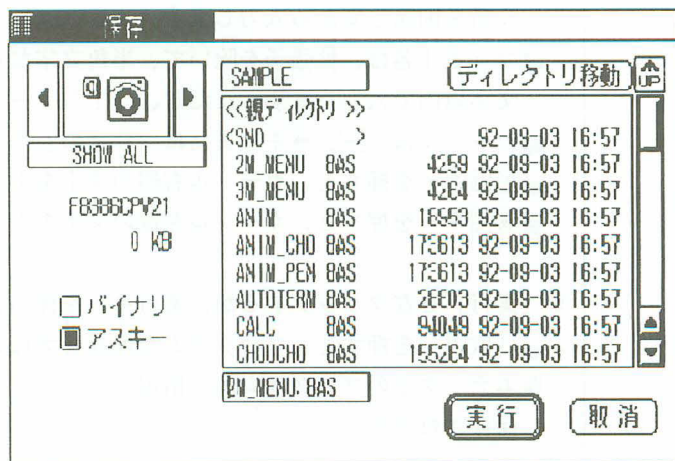
**ご注意**

- 保存せずにF-BASIC386を終了させると、プログラムは消えてしまいます。残しておきたいプログラムは、フロッピィやハードディスクに必ず保存してください。
- 保存先のディスクは、あらかじめFMTownsで使えるように初期化（フォーマット）しておく必要があります。ディスクの初期化については、Townsシステムソフトウェアに添付のマニュアルを参照してください。

**保存の手順**

① ［ファイル］のサブメニューで［保存］を左クリックします。

SHIFT＋CTRL＋Sキーを押しても、同じ結果になります。

次のようなファイルウィンドウが表示されます。

バイナリ：プログラムを内部形式のまま保存する。F-BASIC386以外のテキストエディタなどでは読めなくなる。

アスキー：プログラムをエディタに表示されていた文字のそのままのコードで保存する。F-BASIC386以外のテキストエディタでも表示させて読むことができる。

②ドライブとディレクトリを指定します。

- 現在のドライブ／ディレクトリに保存する場合は、ドライブ／ディレクトリを移動する必要はありません。
- ドライブ／ディレクトリの変更のしかたは、「ファイルウィンドウの操作」（☞ 217ページ）を参照してください。

③保存する形式を、「バイナリ」または「アスキー」で指定します。

④選択ファイル名の枠内にファイル名を入力し、⏎キーを押します。

- 既存のファイルにプログラムを保存する場合は、ファイル名リストの中のファイル名を左クリックします。この場合、実行すると、そのファイルの元の内容は消去されます。
- 選択ファイル名の枠内にすでにファイル名が表示されている場合は、ファイル名を削除してから入力します。
- ファイル名は、拡張子を除いて、半角文字なら8文字以内、全角文字なら4文字以内で入力します。ただし、“！”マークは使えません
- カーソルは、←、→キーまたはマウスの左クリックで左右に移動できます。
- 削除キーを押すと、カーソル右側の文字を1文字削除します。
- 後退キーを押すと、カーソル左側の文字を1文字削除します。

⑤［実行］を左クリックするか、実行キーを押します。

- ［取消］を押すと、プログラムを保存せずにエディタの画面に戻ります。
- エディタ上のプログラムが、指定したドライブの、指定したディレクトリに保存されます。

# 5. プログラムの読み込み

**プログラムの読み込み**

CD-ROMやディスク、フロッピィに入っているプログラムを、エディタの画面に読み込むことができます。メモリ上の元のプログラムは消去されます。
一行実行でLOAD命令を実行しても、同じ結果になります。

**ひとこと**

● 第1部第4章のサンプルプログラムは、この操作でCD-ROMから呼び出せます。また、「2M.BAS」または「3M.BAS」を呼び出して実行すると、サンプルプログラムの一覧からプログラムの名称を指定して実行できます。（☞ 102ページ）

5

**読み込みの手順**

**① ［ファイル］のサブメニューで［読み込み］を左クリックします。**

SHIFT＋CTRL＋L キーを押しても、同じ結果になります。

次のようなファイルウィンドウが表示されます。

読み込み：プログラムを読み込む場合
実行　　：プログラムを読み込んで実行する場合
マージ　：メモリ上のプログラムと読み込んだプログラムを混ぜ合わせる場合

**②ドライブとディレクトリを指定します。**

現在のドライブ／ディレクトリから読み出す場合は、ドライブ／ディレクトリを移動する必要はありません。

ドライブ／ディレクトリの変更のしかたは、「ファイルウィンドウの操作」（☞ 217ページ）を参照してください。

## 5 プログラムの読み込み

③「□読み込み」を指定します。

④ファイル名リストの中から、読み込むファイル名を左クリックします。

⑤［実行］を左クリックするか、実行キーを押します。

［取消］を左クリックすると、何もしないでエディタの画面に戻ります。

プログラムが読み込まれ、エディタの画面に表示されます。

**ご注意**

- オープンされているファイルは、クローズされます。
- エディタに表示されていたプログラムは、プログラムメモリから消去されます。

**ひとこと**

- ファイルウィンドウで「□マージ」を指定すると、メモリ上のプログラムと読み込まれたプログラムが混ぜ合わされます。（☞ 220ページ）
- ファイルウィンドウで「□実行」を指定すると、読み込まれたプログラムが実行されます。（☞ 207ページ）

# 6. ファイルの操作

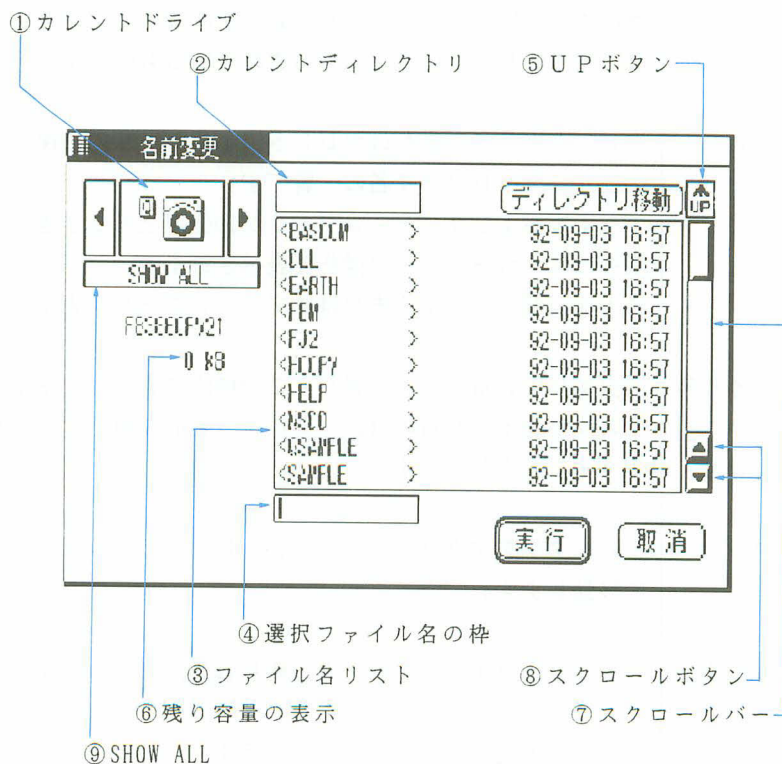
**ファイルの操作**

この節で説明することは、次のとおりです。

- ファイルウィンドウの操作
- ファイル名の変更
- ファイルの削除
- ファイルのマージ
- ディレクトリの作成

**ファイルウィンドウ**

ファイルウィンドウの各部の名称は、次のようになっています。

6

# 6　ファイルの操作

| 項目 | 説明 |
|---|---|
| ①カレントドライブ | 選択されているドライブが表示されます。<br>ドライブはA:～Q:まであります。<br><br>FMTOWNSでは、ドライブ番号は、各装置に次のように割り当てられています。<br>A,B　:本体内蔵のフロッピィディスクドライブ<br>C　　:ROMディスク（ファイルウィンドウでは選択できません）<br>D ～P:ハードディスクドライブ等（接続されている場合）<br>Q　　:CD-ROM |
| ②カレントディレクトリ | 選択されているディレクトリが表示されます。<br>ファイル名リストに＜＞で表示されているものがサブディレクトリです。 |
| ③ファイル名リスト | カレントドライブまたはカレントディレクトリ内に保存されているファイル名およびサブディレクトリ名の一覧です。<br>ファイルおよびサブディレクトリが10個以上あるときは、先頭の10個が表示され、残りはスクロールバーの操作で表示されます。<br>最大1024個のファイルまたはサブディレクトリが表示できます。 |
| ④選択ファイル名の枠 | 選択されているファイル名が、この枠内に表示されます。<br>この枠が表示されるのは、［保存］［名前変更］［ディレクトリ作成］のファイルウィンドウだけです。 |
| ⑤ＵＰボタン | このボタンを左クリックすると、カレントディレクトリが１つ上のディレクトリに移動します。 |
| ⑥残り容量の表示 | カレントドライブのディスクの残り容量が表示されます。 |
| ⑦スクロールバー | ファイル名リストをスクロールします。 |
| ⑧スクロールボタン | ファイル名リストに表示されているファイル名またはサブディレクトリ名を、１行ずつ上下にスクロールします。 |
| ⑨SHOW ALL | 設定されているすべてのドライブを一覧表示します。 |

**ファイルウィンドウの操作**

ファイルウィンドウは、ファイル名を指定するときに使用します。

**ドライブの変更**

A:〜Q:の範囲でカレントドライブを変更できます。変更する方法は次の通りです。

- カレントドライブの両脇の矢印を左クリックして、設定されているドライブを1つずつ移動し、目的のドライブを表示させます。
- [SHOW ALL] を左クリックすると、設定されているすべてのドライブが一覧表示される。目的のドライブを左クリックすると、そのドライブが選択されます。

**ディレクトリの変更**

カレントディレクトリをサブディレクトリ（<>で表示されている）に移動できます。移動する方法は、それぞれ2つずつあります。

1段下の階層のディレクトリに移行する場合

- サブディレクトリをダブルクリックします。
- サブディレクトリを左クリックして、右上の [ディレクトリ移動] を左クリックします。

1段上の階層のディレクトリに移行する場合

- ファイル名リストの最上部の《親ディレクトリ》を左クリックし、右上の [ディレクトリ移動] を左クリックします。
- ファイル名リストの最上部の《親ディレクトリ》をダブルクリックする。
- UPボタンを左クリックします。

**ファイルの選択**

ファイルの選択は、ファイル名リストの中のファイル名を左クリックして行います。[保存] [名前変更] のファイルウィンドウの場合は、選択ファイル名の枠内に、そのファイル名が表示されます。

**ファイル名リストのスクロール**

ファイルおよびサブディレクトリが10個以上あるときは、先頭の10個が表示されます。残りはスクロールバーの操作によって表示することができます。

- スクロールバーの、インジケーターより上の部分を左クリックすると、1ページ分（10個）上にスクロールします。
- スクロールバーの、インジケーターより下の部分を左クリックすると、1ページ分（10個）下にスクロールします。
- インジケーターをマウスでドラッグしても、スクロールできます。
- マウスカーソルをスクロールボタンに合わせて左クリックすると、表示を矢印の方向へ1行ずつスクロールします。

**ファイル名の変更**

プログラムファイルやデータファイルのファイル名を変更します。
一行実行でNAME命令を実行しても、同じ結果になります。

**ご注意**

● 名前を変更するファイルは、クローズされていなければなりません。

**名前変更の手順**

**① ［ファイル］のサブメニューで［名前変更］を左クリックします。**

ファイルウィンドウが表示されます。

**② ドライブとディレクトリを指定し、ファイル名リストから名前を変更するファイルを選択します。**

現在のドライブ／ディレクトリのファイル名を変更する場合は、ドライブ／ディレクトリを移動する必要はありません。

ドライブ／ディレクトリの変更のしかたは、「ファイルウィンドウの操作」（217ページ）を参照してください。

選択ファイル名の枠内に、ファイル名が表示されます。

**③ ファイル名を変更し、↵キーを押します。**

新たに入力するときは、すでに表示されているファイル名を削除してから入力します。

ファイル名は、拡張子を除いて、半角文字なら8文字以内、全角文字なら4文字以内で入力します。ただし、“！”マークは使えません。

カーソルは、←、→キーまたはマウスの左クリックで左右に移動できます。

削除キーを押すと、カーソル右側の文字を1文字削除します。

後退キーを押すと、カーソル左側の文字を1文字削除します。

**④ ［実行］ボタンを左クリックするか、実行キーを押します。**

名前変更をやめたいときは、［取消］を左クリックします。

ファイル名が新しいファイル名に変わります。

## ファイルの削除

プログラムファイルやデータファイルを削除します。
一行実行でKILL命令を実行しても、同じ結果になります。

**ご注意**

- 削除しようとするファイルは、クローズされていなければなりません。
- 一度削除したファイルは、二度と修復できません。必要なファイルを間違って削除しないように、気をつけてください。

削除の手順

**①［ファイル］のサブメニューで［削除］を左クリックします。**

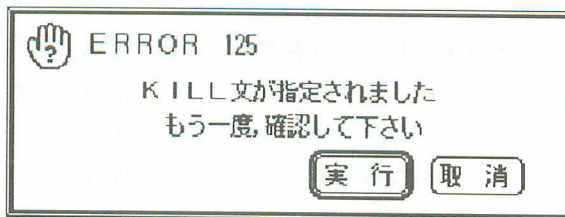
ファイルウィンドウが表示されます。

**②ドライブとディレクトリを指定し、ファイル名リストから削除するファイルを選択します。**

現在のドライブ／ディレクトリのファイルを削除する場合は、ドライブ／ディレクトリを移動する必要はありません。

ドライブ／ディレクトリの変更のしかたは、「ファイルウィンドウの操作」（☞ 217ページ）を参照してください。

**③［実行］ボタンを左クリックするか、実行キーを押します。**

確認のガイドウィンドウが表示されます。

**④ファイルの指定に間違いがなければ、確認のガイドウィンドウの［実行］ボタンを左クリックします。**

指定したファイルが削除されます。

## 6 ファイルの操作

**ファイルのマージ**

プログラムメモリ上にあるプログラムに、指定したファイルのプログラムを混ぜ合わせます。

一行実行でMERGE命令を実行しても、同じ結果になります。

**ご注意**

- マージしようとするファイルは、アスキー形式でなければいけません。
- 指定したファイル中に、メモリ上のプログラムと同じ行番号がある場合、メモリ上のその行は消去されます。

**マージの手順**

**① ［ファイル］のサブメニューで［読み込み］を左クリックします。**

ファイルウィンドウが表示されます。

**②ドライブとディレクトリを指定し、ファイル名リストからマージするファイルを選択します。**

現在のドライブ／ディレクトリのファイルをマージする場合は、ドライブ／ディレクトリを移動する必要はありません。

ドライブ／ディレクトリの変更のしかたは、「ファイルウィンドウの操作」（☞ 217ページ）を参照してください。

**③「□マージ」を指定します。**

**④ ［実行］ボタンを左クリックするか、実行キーを押します。**

マージをやめたいときは、［取消］を左クリックします。

指定したプログラムが読み込まれ、メモリ上にあったプログラムと混ぜ合わされた結果が、エディタの画面に表示されます。

### ディレクトリの作成

サブディレクトリを作成します。

ディレクトリ：プログラムを種類別や目的別、作成者別などに分類して保存したいようなときには、ディレクトリを作成してプログラムを保存します。
ディレクトリを作成すると、ファイルの一覧はディレクトリ単位で表示できます。また、ディレクトリが違えば、同じファイル名でも保存できます。特に、容量の大きなハードディスクでのファイル管理が非常に楽になります。

### 作成の手順

**①［ファイル］のサブメニューで［ディレクトリ作成］を左クリックします。**

ファイルウィンドウが表示されます。

**②ドライブと親ディレクトリを指定します。**

現在のドライブ／ディレクトリに作成する場合は、ドライブ／ディレクトリを移動する必要はありません。

ドライブ／ディレクトリの変更のしかたは、「ファイルウィンドウの操作」（☞ 217ページ）を参照してください。

**③キーボードから作成したいディレクトリ名を入力し、⏎キーを押します。**

ディレクトリ名は、半角文字なら11文字以内、全角文字なら5文字以内で指定します。

**④［実行］ボタンを左クリックするか、実行キーを押します。**

作成をやめたいときは、［取消］を左クリックします。

指定したディスク、あるいはディレクトリ内に新しくディレクトリが作成され、エディタの画面に戻ります。

第3章

# コンパイラを使う前に

この章では、コンパイラをお使いになる前に必要な知識をまとめてあります。

# 1. コンパイラとは

**インタプリタとコンパイラ**

F-BASIC386には、「インタプリタ」と「コンパイラ」の２種類があります。「インタプリタ」と「コンパイラ」では、プログラムの実行に到るまでの過程が異なります。

- コンパイラを使用するためには、F-BASIC386コンパイラ V2.1が必要です。
- コンパイラを効率的に使用するためには、ハードディスクによる運用をおすすめします。
- コンパイラは４MB以上のメモリで運用することをおすすめします。

**インタプリタとは**

インタプリタは、作成したプログラムの命令を１つ１つ、コンピュータが実行できる言葉に翻訳しながら実行していきます。

インタプリタには、次のような長所と短所があります。

長所

- プログラムを作成後、すぐに実行できる。
- プログラムの修正が容易である。

短所

- 命令を翻訳しながら実行するので、実行速度が遅い。

**コンパイラとは**

コンパイラは、作成したプログラムを一度に翻訳（コンパイル）して、コンピュータが直接実行できるプログラムを生成します。そして、生成したプログラムを一気に「実行」します。コンパイラによって生成されたプログラムは、F-BASIC386がなくてもTownsMENU から実行することができます。

コンパイラには、次のような長所と短所があります。

長所

- 翻訳したものを実行するので、実行速度が速い。
- 生成されたプログラムは人間には読むことができないので、秘密の保持に役立つ。

短所

- プログラムを作成後、すぐに実行することができない。
- プログラムの修正がめんどうである。

**インタプリタとコンパイラの互換**

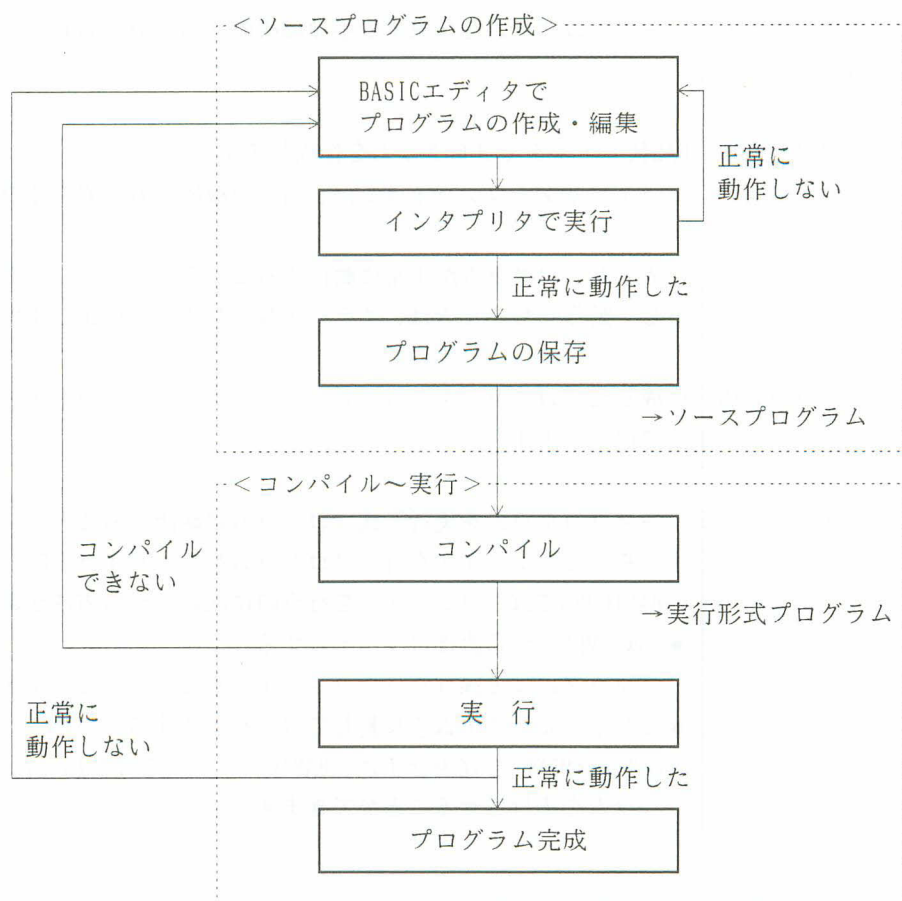
インタプリタで作成したプログラムをコンパイルして、実行形式プログラムを作成することができます。ただし、同じ命令でもインタプリタとコンパイラで機能が異なる場合があります。インタプリタとコンパイラの機能の違いについては、「F-BASIC386 V2.1 リファレンス」を参照してください。

# 2. プログラムの開発手順

2

## プログラムの開発手順

コンパイラは、BASICエディタで作成したプログラムを、FMTOWNSが直接実行できるプログラムにコンパイル（翻訳）します。

コンパイラを利用して、プログラムを開発する手順は次のようになります。
前半の<ソースプログラムの作成>はインタプリタを使用し、後半の<コンパイル～実行>はコンパイラ等を使用します。

## 2 プログラムの開発手順

**ソースプログラムとは**

コンパイラでは、BASICエディタで作成したプログラムのことを**ソースプログラム**と呼びます。ソースプログラムは人間には読むことができますが、コンピュータが直接実行することはできません。コンパイラは、ソースプログラムをもとにコンピュータが実行できるプログラムを生成します。

☞ソースプログラムに対し、コンピュータが直接実行できるプログラムを**実行形式プログラム**といいます。

**ソースプログラムの作成**

ソースプログラムを作成する手順は次の3つに分けられます。

**プログラムの作成・編集**

BASICエディタでプログラムを作成します。
インタプリタとコンパイラでは、命令の機能が若干異なる点に注意が必要です。

**インタプリタで実行**

作成したプログラムが正常に動作することを、インタプリタで確認します。
正しく動作しないときは、エラーがなくなるまで修正します。

**プログラムの保存**

作成したプログラムをフロッピィディスクまたはハードディスクに、**アスキー形式**で保存します。

**コンパイル**

ソースプログラムを実行形式プログラムに翻訳することを、コンパイルといいます。そしてコンパイルを行うプログラムをコンパイラと呼びます。
F-BASIC386では、コンパイルを行うのに次の2つの方法があります。

- TownsMENU から直接コンパイルする。
  ☞マウスによる操作だけでコンパイルすることができます。
- コンソールシステムを起動してコンパイルする。
  ☞TownsMENU を経由せずに、BASICエディタを呼び出したり、実行形式プログラムを実行させることができます。

**実　行**

実行形式プログラムを実行します。
実行形式プログラムは、ソースプログラムファイルと同じディレクトリに作成されています。また、実行形式プログラムは、ソースプログラムファイルと同じ名前で、拡張子が「.EXP」となります。
F-BASIC386では、実行形式プログラムを実行するのに次の２つの方法があります。

- TownsMENU から直接実行する。
  TownsMENU で実行形式プログラムを選択し、実行させます。
- コンソールシステムから実行する。
  コンソールシステムで、コマンド（RUN386）と実行形式プログラム名を入力し、⏎キーを押します。

**プログラムの修正**

プログラムが正常に動作しないときは、BASICエディタを呼び出してソースプログラムを修正・保存し、再びコンパイルします。コンパイラで正しく動作するまで、コンパイル→実行→修正を繰り返します。

# 3. ハードディスクの利用

**コンパイラはハードディスクで使う**

F-BASIC386コンパイラは、ハードディスクで運用することをおすすめします。コンパイラは、非常にたくさんの処理を行います。したがって、ソースプログラムのサイズが大きくなればなるほど、コンパイルにかかる時間も長くなります。コンパイルを効率的に行うためにも、F-BASIC386コンパイラをハードディスクから起動してコンパイルを行ってください。

コンパイラをハードディスクで使用するには、ハードディスクに最低10MBの空き領域が必要です。ただし、実際には4MB以上の空き領域を確保して運用することをおすすめします。

F-BASIC386のシステムをハードディスクから使えるようにするには、以下の作業が必要です。

①ハードディスクの区画とドライブ番号の設定
②CD-ROMから［HD INSTALL］の実行
③起動バッチファイル（AUTOEXEC.BAT）の変更

これらの作業の具体的な手順については、☞16ページを参照してください。

# 4. コンパイラとTownsOS V1.1

**コンパイル時の注意事項**

F-BASIC386コンパイラV2.1で作成した実行形式プログラムは、TownsOS V1.1では実行できません。
TownsOS V1.1で実行できるプログラムを作成するには、コンパイラのメニューバーの設定を「V1.1」にしてからコンパイルする必要があります。
設定方法については235ページを参照してください。

**作成時の注意**

● TownsOS V1.1では、以下の命令・機能は使用できません。
DEF FONT（マルチ・ベクトル・12 dotフォント）
IF～THEN, ELSE, ENDIF（block if文）
LOAD@, SAVE@での圧縮TIFF対応
LOAD@, SAVE@でのJPEG対応
LOAD@ でのTIFFファイルおよび音色ファイルの配列への読み込み
OPENでのALLOW モード指定
PLAY@（EUPファイルの演奏）
SCREEN（グラフィック２画面モード）
MOVIE （動画の再生）
SMSGPLAY（サウンドメッセージの再生）
CLEAR でのDLL 領域の指定
MOUSE 0 でのマウスカーソル描画ページの指定
MOUSE 6
ハードコピーの48ドットプリンタ対応
ESC/P

F-BASIC386コンパイラのV1.1とV2.1では、使用するライブラリ等が異なります。
High-C等開発環境をすでにお持ちの方は、特に注意してください。

## 4　コンパイラとTownsOS V1.1

**実行時の注意**

- TonwsOS V1.1で実行できる実行形式プログラムを実行する時には、以下の点に注意してください。
  - TownsOS V2.1で実行するときには、かならずアイテム登録でV1.1アプリと指定してください。
  - CPUにi486™SXを使用しているマシンで実行するときには、動作モードを互換モードに切り換えてください。
  - TonwsOS V1.1で実行できる実行形式プログラムはLANには対応していません。
  - V1.1のシステム標準のPCM 音色ファイルを使用する場合は、コンパイラのCDの¥BASCOM¥V11 のディレクトリから“DRUMS.PMB” を、作成した実行形式プログラムのあるディレクトリにコピーしてください。

**プログラムの見分け方**

コンソールモード、あるいはコマンドモードで次のように入力すると、使用したコンパイラについての情報が表示されます。

　　type ～.exp⏎　（～には、実行形式ファイル名を入力します。）

# 第4章

# コンパイラの操作

この章では、コンパイラの操作方法をプログラムの開発手順に添って説明しています。

F-BASIC386では、次の２つの方法でコンパイルを行うことができます。

　TownsMENU から直接コンパイルおよび実行する

　コンソールシステムを起動してコンパイルおよび実行する

ソースプログラムを作成する際の注意点を説明したあと、それぞれの操作方法について解説します。続いて、コンパイル時の条件の変更のしかたと、コンパイル中に表示されるメッセージについて説明します。

# 1. ソースプログラムの作成

**ソースプログラムの作成**

コンパイルするときの元になるソースプログラムは、BASICエディタで作成します。

BASICエディタの操作については、185ページを参照してください。

**インタプリタとの互換性**

インタプリタとコンパイラでは、一部、機能が異なります。このため、コンパイラで生成された実行形式プログラムの動作と、ソースプログラムをインタプリタで実行した場合の動作が異なる場合があります。

**機能の異なる命令**

インタプリタとコンパイラでは、命令の機能に次に示すような相違点があります。

**使用できない命令**

次に示す命令はコンパイラでは使用できません。

CONT, DELETE, LIST, LLIST, LOAD, MERGE, NEW, RENUM, SAVE

**機能の異なる命令**

次に示す命令の機能は、インタプリタと若干異なります。

CHAIN, COMMON, CLEAR, ERL 関数, FRE関数, MOUSE関数, RUN, STOP, TRON

**変数・関数を宣言する命令**

次に示すDEF命令による宣言は、実行順序に関係なく、ソースプログラム上の述順序にしたがって有効になります。

DEF FN, DEFINT, DEFLNG, DEFSNG, DEFDBL, DEFSTR

**ひとこと**

インタプリタとコンパイラにおける、命令の機能の具体的な相違点については、「F-BASIC386 V2.1 リファレンス」で各命令を参照するか、同「付録6　インタプリタとコンパイラの機能の違い」をご覧ください。

**処理速度**

インタプリタと比べ、コンパイラは非常に速くプログラムを実行します。このため、FOR～NEXTループなどでタイミングをとっている場合は、値の変更が必要になります。

INTERVAL, TIME$, WAIT命令を使用すれば、値の変更は必要ありません。

**ソースプログラムの保存**

作成したソースプログラムは、フロッピィディスクまたはハードディスクに保存しておきます。

保存の形式は、**アスキー形式**を指定します。

ファイル名は半角文字で入力します。ただし、“！”マークは使えません。

プログラムの保存のしかたは、211ページを参照してください。

# 2. TownsMENU からのコンパイルと実行

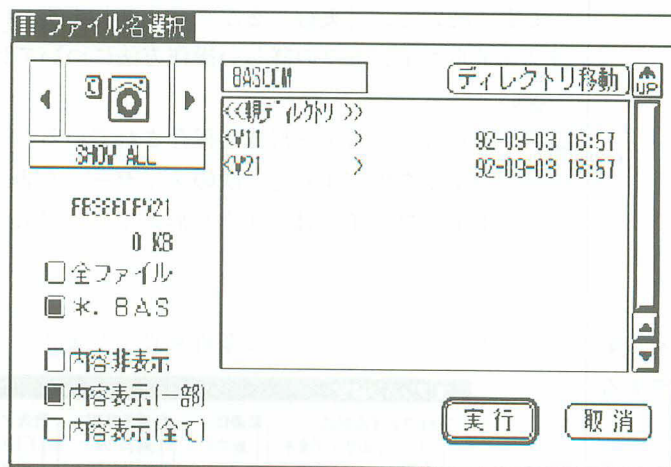
**TownsMENU からのコンパイル**

F-BASIC386コンパイラでは、TownMENUからマウスによる操作だけでコンパイルを行うことができます。

F-BASIC386コンパイラでは、コンソールシステムでコンパイルすることもできます。コンソールシステムの使い方については239ページを参照してください。

**1 [COMPILER] を実行する**

ディスクフォルダ内の［COMPILER］を選択し、実行します。

次のようなコンパイラのファイルウィンドウが表示されます。

- ［全ファイル］を左クリックすると、指定されているディレクトリ内にあるすべてのファイル名が表示されます。
- ［＊．ＢＡＳ］を左クリックすると、指定されているディレクトリ内にある、サブディレクトリと、拡張子が「.BAS」のファイル名が表示されます。
- ［内容非表示］を左クリックすると、ファイル選択後、ファイルの内容は表示されません。
- ［内容表示（一部)］を左クリックすると、ファイル選択後、ファイルの内容の先頭部分が表示されます。（約４Ｋバイト分）
- ［内容表示（全て)］を左クリックすると、ファイル選択後、ファイルの内容がすべて表示されます。

ご　注　意

● [COMPILER]を実行すると、<BASCOM>ディレクトリ内のファイル“BC.BAT”と<BASCOM¥V21>ディレクトリ内のファイル“CONSBAS.EXE”が実行されます。<BASCOM>ディレクトリ内の“BC.BAT”と<BASCOM¥V21>ディレクトリ内の“CONSBAS.EXE”がないと、[COMPILER]を実行できなくなります。この2つのファイルは名前を変更したり、削除したりしないでください。

**2 ファイルを選択する**

ファイル名一覧から、コンパイルするソースプログラムファイル名を左クリックします。続いて、［実行］を左クリックします。

ファイルウィンドウの詳しい操作方法については217ページを参照してください。

ファイルは、アスキー形式で保存されているものを指定します。バイナリ形式のファイルを指定すると、次のメッセージを表示して処理を中断します。

「指定のファイルはバイナリ形式です　ASCII形式でSAVEしてください
［確認］」

**3 コンパイル条件を設定する**

ウィンドウ内で、コンパイル条件を設定します。

● 実行ファイル形式

ファイルサイズ優先…サイズは小さいけれども、実行速度の遅い実行形式プログラムが生成されます。

実行スピード優先……サイズは大きいけれども、実行速度の速い実行形式プログラムが生成されます。

● 最適化

する……使用してしない変数や空きのループを取り除いた実行形式のプログラムが生成されます。

しない…コンパイルにかかる時間が短くなります。

●行番号情報

持つ………行番号情報を持った実行形式ファイルが生成されます。実行時にエラーが起こったときは、エラーの起きた行番号が表示されます。

持たない…行番号情報を持たない実行形式ファイルが生成されます。実行形式プログラムのサイズは小さくなります。実行時にエラーが起きたときに、エラーの起きた行番号は「0」とだけ表示されます。

●警告メッセージ

表示する……プログラムミスの可能性があるときに、警告メッセージが表示されます。

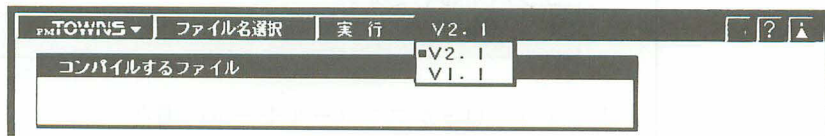
表示しない…プログラムミスの可能性があっても、警告メッセージは表示されません。

●エラー発見時

続ける……コンパイル時にエラーを発見しても、コンパイラは処理を続けます。

中断する…コンパイル時にエラーを発見すると、コンパイラは処理を中断します。

**4 コンパイルバージョンを選択する**

メニューバーで、F-BASIC386のV2.1と、V1.1のどちらでコンパイルするかを選択します。

●[V2.1]を選択すると、TownsOS V2.1だけで実行できる実行形式プログラムが生成されます。「F-BASIC386 V2.1 リファレンス」に載っているすべての命令・機能が使用できます。[V2.1]を選択して作成したプログラムは、TownsOS V1.1では実行できません。

●[V1.1]を選択すると、TownsOS V1.1で実行できる実行形式プログラムが生成されます。F-BASIC386 V2.1 の命令・機能の中に、一部、使用できないものがあります。

F-BASIC386 V1.1 では実行できない機能・命令については、☞ 229ページを参照してください。

**5 コンパイルを開始する**

メニューバーの［実行］を左クリックします。

指定したファイルと同じ名前の実行形式ファイルが既にあるときは、次のメッセージが表示されます。

「指定のファイルの実行ファイルが存在します
コンパイルを実行してもよろしいですか　　　［実行］「取消］」

［実行］を左クリックすると、コンパイルが開始されます。同じ名前のファイルは新しいファイルに置き換えられます。

［取消］を左クリックすると、中断します。

次のような画面に切り替わり、コンパイルが開始されます。

ＴｏｗｎｓＯＳがコンソールレスモードの場合

```
ＢＡＳＩＣコンパイラ　コンソールシステム

CONSBAS.EXE Version 1.2
Copyright (C) FUJITSU LIMITED 1987-1992. All right reserved.
ＰＦ１１／１２キーで画面モードが切り替わります

A>:echo off
コンパイルします。
　　　：
```

ＴｏｗｎｓＯＳがコンソールモードの場合

```
A>:echo off
コンパイルします。
　　　：
```

「コンパイルします。」に続いて、コンパイル処理の経過を示すメッセージが次々に表示されます。各メッセージの詳しい意味については、246ページを参照してください。

続いて、次のメッセージが表示されます。

```
リンクをします。
TLINK: 2.2d -- Copyright (C) 1986-89 Phar Lap Software, Inc.
```

コンパイラ内部の処理は、「コンパイル」と「リンク」に分けられます。両方の処理が正常に終わってはじめて、実行できるプログラムが生成されます。

F-BASIC386の使用環境や、ソースプログラムの量によって、「リンク」に非常に時間がかかる場合があります。ディスクまたはCD-ROMのアクセスランプが点灯しているうちは処理中です。そのままお待ちください。

「リンク」が終了すると、次のメッセージが表示されます。

```
準備ができたらどれかキーを押してください。
```

**6 コンパイルを終了する**

何かキーを押します。

この場合はスペースキーを押すのが無難です。

画面はTownsMENU に戻ります。

**ご注意**

コンパイル中にエラーが発生した場合は、次のメッセージが表示され、処理が中断されます。

```
エラーが発生しました
準備ができたらどれかキーを押してください.
```

このメッセージの直前に表示されたエラーメッセージで、エラーの内容を確認のうえ、何かキーを押します。

画面はTownsMENU に戻ります。

エラーメッセージ内容に応じて、BASICエディタでソースプログラムを修正し、再度コンパイルを行なってください。

各エラーメッセージの意味と処置方法については、☞ 247ページを参照してください。

## 2 TownsMENU からのコンパイルと実行

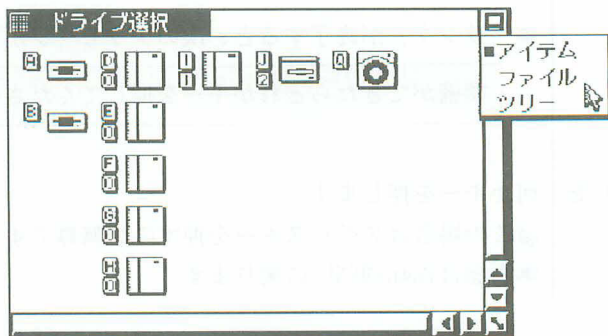
**TownsMENU からの実行**

コンパイルによって生成された、実行形式プログラム（拡張子「.exp」）を実行させます。

- コンソールシステムから実行形式プログラムを実行させる方法については、☞ 242ページを参照してください。

実行形式プログラムは、次のように実行させます。

**1 表示モードをファイルにする**

ドライブ選択ウィンドウの表示切替えボタンを左クリックし、次に「ファイル」を左クリックします。

**2 ファイルを表示する**

実行するファイルが保存されているドライブのドライブアイコンを左クリックし、「ファイル」メニューの「実行／開く」を選ぶと、ファイル表示ウィンドウが表示されます。

- 実行するファイルが保存されているドライブのドライブアイコンをダブルクリックしても、同じ結果になります。

**3 プログラムを選択し、実行する**

実行したい実行形式プログラムファイルを選択し、実行します。

- コンパイルによって生成された実行形式プログラムは、ソースプログラムと同じ名前で、拡張子だけが「.exp」となって保存されています。保存されているディレクトリは、ソースプログラムと同じです。
- 必要に応じてアイテム登録をしておくと便利です。アイテム登録のしかたについては、「Townsシステムソフトウェア」に添付のマニュアルをご覧ください。

- プログラムの実行が終了すると、TownsMENU に戻ります。

# 3. コンソールシステムからのコンパイルと実行



## コンソールシステム

F-BASIC386コンパイラには、コンパイラ用のコンソールシステムが用意されています。コンソールシステムでは、TownsMENUを経由せずに、BASICエディタを呼び出したり、コンパイルしたプログラムを実行させたりすることができます。

TownsMENUから直接、マウスによる操作だけでコンパイルすることもできます。TownsMENU から行うコンパイルについては233ページを参照してください。

コンソールシステム上の操作を、次の４つに分けて説明します。

**コンソールシステムの起動**

**コンパイル**

**実行形式プログラムの実行**

**コンソールシステムの終了**

## コンソールシステムの起動

コンソールシステムを起動するときは、TownsMENU でアイテム［コンソール］を実行します。

**［コンソール］を実行する**

ディスクフォルダ内の［コンソール］を選択し、実行します。

次のようなコンソールシステムの画面に切り替わります。

```
ＢＡＳＩＣコンパイラ　コンソールシステム
CONSBAS Version 1.2
Copyright (C) FUJITSU LIMITED 1987-1992. All right reserved.
ＰＦ１１／１２キーで画面モードが切り替わります。

D>
```

**PF11キー** 画面モードを切り替えます。

**PF12キー** 画面モードをコンソールシステムのモードに切り替えます。

## 3　コンソールシステムからのコンパイルと実行

**コンソールシステムからのコンパイル**

コンソールシステムでコンパイルするときは、コンパイラを起動するコマンド"bc"と、ソースプログラムファイル名を直接入力します。

**1 カレントディレクトリを移動する**

カレントディレクトリを、ソースプログラムのあるディレクトリに移動します。

例）Eドライブのルートディレクトリにある、<BASIC>ディレクトリの下の、<BASAPL>ディレクトリにソースプログラムがあるとき

```
CD E:¥BASIC¥BASAPL⏎
```

ディレクトリの詳しい移動方法については、☞264ページを参照してください。

**ご注意**

コンソールシステムをCD-ROMから起動している場合は、次の命令も実行してください。　PATH Q:¥BASCOM¥V21⏎

この命令はコンソールシステムを終了し、TownsMENUに戻ると無効になります。

**2 コンパイルを開始する**

プロンプトに続けて、次のように入力し、⏎キーを押します。

```
bc ソースプログラムファイル名
```

**ソースプログラムファイル名**

- アスキー形式で保存されているソースプログラムファイル名を記述します。
- 拡張子は省略します。
- 指定したいソースプログラムファイルがカレントにないときは、ディレクトリを移動するか、パス名を入力します。

パス名の入力のしかたは、☞「Townsシステムソフトウェア」に添付のマニュアルご覧ください。

ディレクトリの移動のしかたは、☞264ページを参照してください。

**ご注意**

- 指定したファイルと同じ名前の実行形式ファイルが、ソースプログラムファイルと同じディレクトリに存在しているときは次の点に注意してください。
  実行形式ファイルは、コンパイルによって生成されたファイルに置き換えられます。このとき、コンソールシステムでは特にメッセージを出しません。すでに存在しているファイルが必要なものの場合は、別のディレクトリに複写するか、ファイル名を変更してからコンパイルを行ってください。

- コンパイラを起動するときのコマンドファイル“BC.BAT”は、名前を変更したり、削除したりしないでください。“BC.BAT”がないと、TownsMENU から直接行うときのアイテム[COMPILER]が実行できなくなります。これは、[COMPILER]が“BC.BAT”を呼び出すようになっているためです。

次のメッセージが表示され、コンパイルが開始されます。

```
コンパイルします。
```

続いて、コンパイル処理の経過を示すメッセージが次々に表示されます。各メッセージの詳しい意味については、246ページを参照してください。

続いて、次のメッセージが表示されます。

```
リンクをします。
TLINK: 2.2d -- Copyright (C) 1986-89 Phar Lap Software, Inc.
```

コンパイラ内部の処理は、「コンパイル」と「リンク」に分けられます。両方の処理が正常に終わってはじめて、実行できるプログラムが生成されます。

F-BASIC386の使用環境や、ソースプログラムの量によって、「リンク」に非常に時間がかかる場合があります。ディスクまたはCD-ROMのアクセスランプが点灯しているうちは処理中です。そのままお待ちください。

「リンク」が終了すると、次のメッセージが表示されます。

```
準備ができたらどれかキーを押してください。
```

**3 コンパイルを終了する**

何かキーを押します。

この場合はスペースキーを押すのが無難です。

画面はプロンプトの状態に戻ります。

**ひとこと**

コンパイルによって生成された実行形式ファイルは、ソースプログラムと同じ名前で拡張子だけが「.exp」となり、ソースプログラムファイルと同じディレクトリに保存されます。

## 3 コンソールシステムからのコンパイルと実行

**ご注意**

コンパイル中にエラーが発生した場合は、次のメッセージが表示され、処理が中断されます。

```
エラーが発生しました
準備ができたらどれかキーを押してください.
```

このメッセージの直前に表示されたエラーメッセージで、エラーの内容を確認のうえ、何かキーを押します。

- 画面はプロンプトの状態に戻ります。
- エラーメッセージ内容に応じて、BASICエディタでソースプログラムを修正し、再度コンパイルを行なってください。
- コンソールシステムから直接BASICエディタを呼び出す方法については、243ページを参照してください。
- 各エラーメッセージの意味と処置方法については、247ページを参照してください。

**コンソールシステムからの実行**

コンパイルによって生成された実行形式プログラムを、コンソールシステムから実行します。

実行形式プログラムは、次のように実行します。

- PATH命令でパスが設定してあるときは、次のように入力し、パスを解除しておく必要があります。 PATH; ⏎

**実　行**

プロンプトに続けて次の形式で入力し、⏎キーを押します。

```
RUN386　実行形式プログラム名
```

**実行プログラム名**

- 「.EXP」形式のファイルを指定します。このとき、拡張子は省略できます。
- 実行形式プログラムが、"RUN386"と異なるディレクトリにあるときは、パス名を入力します。

- 実行後、PF12キーを押すとコンソールシステムの画面に戻ります。

## 3　コンソールシステムからのコンパイルと実行

**ひとこと**

コンパイルによって生成された実行形式プログラムは、ソースプログラムと同じ名前で、拡張子だけが「.exp」となって保存されています。保存されているディレクトリは、ソースプログラムと同じです。

**ご注意**

次の形式のRUN 命令を含むプログラムは、コンソールシステムからは実行できません。TownsMENU から実行してください。

RUN　ﾌｧｲﾙﾃﾞｨｽｸﾘﾌﾟﾀ

**コンソールシステムの終了**

コンソールシステムからは、TownsMENUに戻ることも、BASICエディタを呼び出すこともできます。

コンパイル中、および実行中にエラーが発生したときは、直接BASICエディタを呼び出し、ソースプログラムを修正した方が効率的です。

**TownsMENUに戻る**

プロンプトに続けて次のように入力し、⏎キーを押します。

```
exit
```

しばらくすると、画面はTownsMENU に戻ります。

**BASIC エディタを呼び出す**

プロンプトに続けて次のように入力し、⏎キーを押します。

"FB386" または"FB386M"のあるディレクトリをカレントディレクトリにしてから、実行します。

```
RUN386 FB386[M]
```

"FB386" を指定すると、オンラインマニュアルは使えません。

"FB386M"を指定すると、オンラインマニュアルが使えます。

しばらくすると、画面はBASICエディタになります。

**ひとこと**

コンソールシステムから直接呼び出したBASICエディタを終了させたあと、PF12キーを押すと、コンソールシステムの画面に戻ります。

# 4. コンパイル時の条件の変更

**コンパイル時の条件の変更**

コンパイラを起動させるコマンドファイル“BC.BAT”の内容を書き換えるとコンパイルの際の条件を変更することができます。

- “BC.BAT”の内容の変更は、TownsMENU からのコンパイルとコンソールシステムからのコンパイルの、どちらの方法でコンパイルを行っても、有効になります。
- “BC.BAT”の内容は、TownsMENUシステムソフトウェアの「テキスト編集」で変更します。

**“BC.BAT”を書き換える**

コンパイル時の条件を変更するときは、“BC.BAT”の中の次の行を書き換えます。

```
q:¥run386 q:¥bascom¥V21¥fb386cp -speed -w0 …
```

- [HD INSTALL]を実行してハードディスクに複写された“BC.BAT”では、ドライブ名(q:)や、パス名（¥bascom)が異なります。

この行の「-speed -w0」の箇所を、必要に応じて書き換えます。

例）d:¥run386 d:¥basic¥bascom¥V21¥fb386cp -noopt …

指定できる文字列を以下に示します。

- 書き換えるときは、必ず半角文字の英小文字で入力します。

**-e**

指定する　コンパイラはエラーを発見した時点でコンパイル処理を中断します。

指定しない　コンパイラはエラーを発見してもコンパイル処理を続けます。

**-noopt**

指定する　最適化処理が行われません。コンパイルにかかる時間は短くなります。

指定しない　最適化処理が行われます。コンパイルにかかる時間は長くなりますが、生成される実行形式ファイルは高速に実行されるものになります。

| オプション | 説明 |
|---|---|
| -speed,<br>-space | -spaceを指定する　ｐコードの実行形式ファイルが生成されます。ｐコードの実行形式ファイルは、サイズが小さくなりますが、実行速度は遅くなります。<br>-speedを指定する　ネイティブコードの実行形式ファイルが生成されます。ネイティブコードの実行形式ファイルは、サイズが大きくなりますが、実行速度は速くなります。<br>初期値は-spaceになっています。<br>-speedと-spaceの両方を指定したときは、後ろで指定した方が有効になります。 |
| -nodebug | 指定する　行番号情報を持たない実行形式ファイルが生成されます。実行形式ファイルのサイズが小さくなります。ただし、実行時にエラーが起こっても、エラーが起きた行番号は「０」としか表示されません。<br>指定しない　行番号情報を持った実行形式ファイルが生成されます。実行時にエラーが起こったときに、エラーの起こった行番号が表示されます。 |
| -w0, -w1 | -w0 を指定する　コンパイル時に文法的に怪しい点があっても、警告メッセージは表示されません。<br>-w1 を指定する　コンパイル時に文法的に怪しい点があると、警告メッセージが表示されます。<br>初期値は-w1 になっています。<br>-w0 と-w1 の両方を指定したときは、後ろで指定した方が有効になります。 |
| -asm | 指定する　.asmファイルが生成されます。.asmファイルは、アセンブル（386ASM）するときのもとになるテキストファイルです。<br>指定しない　.objファイルが生成されます。.objファイルは、リンクするときのもとになるバイナリ形式のファイルです。<br>"BC.BAT"の中で、-asmを指定することはできません。-asmは「コンパイル」と「リンク」を別々に行うときに指定できます（266ページ参照）。 |

# 5. コンパイル中に表示されるメッセージ

**コンパイル中のメッセージ**

コンパイル中には、処理の経過を示すメッセージがいろいろ表示されます。そのメッセージの意味を以下で説明します。
コンパイル時の条件を変更せず、問題なくコンパイルが終了したときは、次のようなメッセージが表示されます。

```
F-BASIC386 Compiler Version 2.1 Level 10
Copyright (C) FUJITSU LIMITED 1992
Developed by FUJITSU PERSONAL COMPUTER SYSTEMS LIMITED
Basic Code Analyzer.                ………… ﾌﾟﾛｸﾞﾗﾑ を解析中
Optimizer.                          ………… 最適化中
80386 Native Code Generator.        ………… 386 のｺｰﾄﾞ に変換中
Jump Fixup.                         ………… 分岐先を確認中
Emitter.                            ………… 中間ﾌｧｲﾙ生成中
No warning No error detected.       ………… エラー情報
```

- エラーがあるときは、「Basic Code Analyzer.」に続いてエラーメッセージが表示されます。各エラーメッセージの意味については、☞ 247ページを参照してください。
- 警告があるときは、「Optimizer.」に続いて警告メッセージが表示されます。各警告メッセージの意味については、☞ 256ページを参照してください。
- エラーおよび警告があった場合には、「No warning No error detected.」のところに、エラーと警告の個数がそれぞれ表示されます。
  例) 1 warning(s) 2 error(s) detected.
- コンパイル中のメッセージは、コンパイル時の条件や、ソースプログラムの内容によって多少異なります。
  たとえば、CHAIN 命令を含むプログラムをコンパイルした場合は、「Chain Program.」というメッセージに続いて、連結されたプログラムの数だけ「Basic Code Analyzer.」～「80386 Native Code Generator.」のメッセージが繰り返し表示されます。

# 第5章

## コンパイラのエラーメッセージ

コンパイラでは、コンパイル時とプログラムの実行時にエラーが発見され、エラーメッセージが表示されます。また、プログラムミスの可能性があるときには、警告メッセージが表示されます。この章では、コンパイラを使用した場合に表示されるエラーメッセージについて、次の3つに分けて意味と対処方法を説明します。

- コンパイル時のメッセージ
- 警告メッセージ
- 実行時のメッセージ

# 1. コンパイル時のメッセージ

**コンパイル時のエラー**

コンパイル時にエラーを発見した場合、コンパイラはメッセージを表示して処理を中断します。エラーメッセージは次の形式で表示されます。

ソースプログラム名（行番号）：error:メッセージ

☝コンパイル時のパラメータに“-e”を指定したときは、エラーを発見しても処理を続けます。

コンパイル時に表示されるエラーには次のものがあります。

**0で割ることはできません**

原因：除算の除数が0です。

対処：定義されていない変数を除数に使用しないようにしてください。

**COMMONの使いかたが正しくありません**

原因：呼出し元と呼出し先で、COMMONの内容が異なります。

対処：呼出し元と呼出し先のCOMMON命令の内容を同じものにしてください。

**DEF SPRITE: DEF SPRITEに続く数字が異常です**

原因：DEF SPRITEの引数が正しくありません。

対処：指定された引数が、指定できる範囲の数値を超えていないかチェックしてください。

**FNxxx: 再帰呼出しは使えません**

原因：ユーザー定義関数の式中にユーザー定義関数が使われています。

**FNxxx: 引数が二重定義されています**

原因：ユーザー定義関数に同じ名前の引数が使用されています。

**FNxxx: 引数の数が多すぎます**

原因：このユーザー定義関数を呼ぶための引数が多すぎます。

**IFブロックにENDIF 文がありません**

原因：構造化IF命令に対応するENDIF 命令がありません。

対処：ENDIF 命令があるかどうか確認してください。

**IFブロックの構造が複雑すぎます**

原因：IFブロックの入れ子の数が127 を超えています。

対処：入れ子の数を確認してください。

**IFブロックの構成に誤りがあります**

原因：IFブロックの命令の記述位置に誤りがあります。

対処：構造化ブロックの構成が正しいか確認してください。

**RUN file, R: この機能は現システムでは処理できません**

原因：RUN 命令で<R>指定を行おうとしました。

対処：コンパイラでは、 RUN命令で<R>指定を行うことはできません。問題がないようならば、この命令を削除してください。どうしても、この命令が必要ならばコンパイルをあきらめ、ソースプログラムをインタプリタで実行してください。

**RUN label: この機能は現システムでは処理できません**

原因：RUN命令で実行するプログラム名に、ラベルまたは行番号を指定しようとしました。

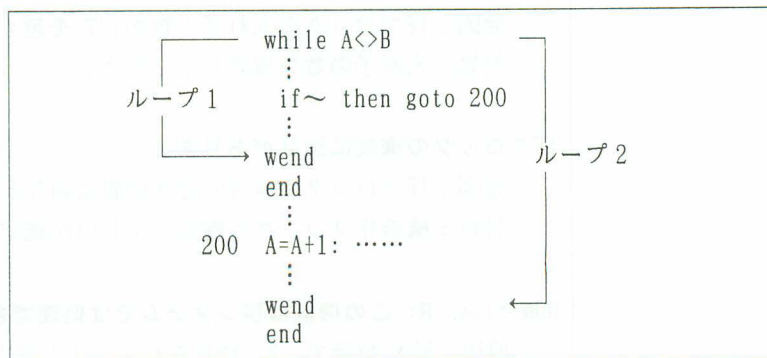
対処：コンパイラでは、RUN の後にラベルまたは行番号を指定することはできません。問題がないようならば、この命令を削除してください。どうしても、この命令が必要ならばコンパイルをあきらめ、ソースプログラムをインタプリタで実行してください。

**WHILE～WENDループの数が多すぎます**

対処：WHILE～WENDループの数を減らしてください。

# 1　コンパイル時のエラーメッセージ

**WHILE文がないのにWEND文がありました**

対処：次のようなプログラム構成になっていないかチェックしてください。

**WHILE 文に対応するWEND文がありません**

原因：WHILE 文に対応するWEND文がありません。

対処：２つのWHILE 文に対し、１つのWENDで対応させたりしないようにしてください。

**英字の順番が間違っています**

原因：DEFINT等の宣言が間違っています。

**演算中に文字列が存在しています**

対処：文字変数を演算中に使用しないようにしてください。

**漢字コードが正しくありません**

原因：許される範囲を超えた値の漢字コードが指定されています。

### 関数または命令文の使い方が正しくありません

原因：パラメータの設定または値が違います。

対処：・パラメータの範囲を合わせてください。
・パラメータの使用する数の形式が正しいか確認してください。
・宣言されていない配列や使用されていない変数を使わないようにしてください。(GET@, PUT@ 命令等)
・指定された配列の大きさが小さくないかチェックしてください。(GET@, PUT@ 命令等)
・配列の添字の値は正にしてください。
・PRINT USING 命令の桁指定は24桁までにしてください。

### 行番号が許される範囲を超えています

原因：行番号が 1〜65529 の範囲にありません。

対処：行番号は 1〜65529 の範囲の整数を使用してください。

### コードが正しくありません

原因：許される範囲を超えた値のキャラクタコードが指定されています。

### 作業用のファイルが作成できません

原因：指定されたディスク装置が使用可能な状態になっていません。

対処：・指定した装置が正しく接続されているか確認してください。
・フロッピィディスクが正しくセットされているか確認してください。

**数値データの値が許される範囲を超えています**

原因：変数型式（単精度、倍精度、整数）以上に演算結果が大きくなってしまいました。

対処：・整数演算の結果が－32768 ～＋32767 の範囲にあるようにしてください。

・ロング型整数演算の結果が－2147483648～＋2147483647の範囲にあるようにしてください。

・実数演算などの結果が、単精度のときは－3.40282E＋38～＋3.40282E＋38、倍精度の時は－1.797693134862 31D＋308 ～＋1.79769313486231D＋308 の範囲にあるようにしてください。

☞整数演算や実数変数の演算における範囲外の演算についてはプログラム実行時に検出されます。

**存在しない行番号が参照されています**

原因：存在しない行番号をプログラム中で指定しています。

対処：GOTO, GOSUB などにおいて、指定する行番号がプログラム中に存在するか確認してください。

**第２引数が定数になっていません**

原因：この関数の第２引数は定数でなければなりません。

**ディスクに空き領域がありません**

対処：不要なファイルを削除するか、または容量の残っている他のフロッピィディスクを使用してください。

☞空き領域は、FILES命令によって知ることができます。

**ディスクの書き込みに失敗しました**

原因：指定されたディスク装置が使用可能な状態になっていません。

対処：・指定した装置が正しく接続されているか確認してください。

・フロッピィディスクが正しくセットされているか確認してください。

**倍精度（変数）は使用できません**

原因：ここでの倍精度（変数）の使用は許可されません。

**引数が定数になっていません**

原因：この関数の引数は定数でなければなりません。

**引数の数が足りません**

原因：ユーザ定義関数の引数の数が足りませんでした。

**ファイル内に直接実行文があります**

原因：ソースプログラム中に行番号のない命令があります。

対処：・単なるデータファイルを間違って読み込んでいないか確認してください。

・BASICエディタ以外のエディタで作成したプログラムのときはすべての行に行番号があるか確認してください。

**ファイル名: 指定のファイルが見つかりません**

原因：存在しないファイルを指定した。

対処：・正しいファイル名を指定してください。

・ドライブ番号の指定が正しいか確認してください。

**文法が正しくありません**

原因：・プログラムの中にF-BASIC386にない命令があります。

・命令が正規の形式以外で使用されています。

対処：・命令の綴りを修正してください。

・関数を代入文の左辺においたり、命令として使用しないでください。

・変数名は英字または漢字で始めてください。

・ERR, ERL, CSRLINの予約変数に値を代入しないでください。

・行番号は65529 までの正の整数にしてください。

・IF命令中でELSEを使用する場合は必ずTHENをおいてください。

・パラメータの数を合わせてください。

・DEF FN命令で定義した関数の使い方を確認してください。

・ラベル名の先頭は必ず*(アスタリスク) を指定してください。

・左と右のカッコの数を合わせてください。

## 1　コンパイル時のエラーメッセージ

**変数：変数または式の型が合っていません**

原因：・数値を文字変数に代入しようとしました。
　　　・文字を数値変数に代入しようとしました。

対処：・数値を文字変数に代入しないようにしてください。
　　　・文字を数値変数に代入しないようにしてください。
　　　・関数の引数の型に合わせてください。
　　　・FOR文での制御変数に倍精度変数を使用しないようにしてください。

**変数：重複定義を行おうとしました**

原因：COMMON文で指定した変数名は、すでに別のCOMMON文で定義されています。

対処：どちらかの変数名を修正するか、削除してください。

**変数：同名の関数が定義されています**

原因：この変数名と同じ名前のユーザー定義関数が既に宣言されています。

対処：一方の変数名を変更するか、削除してください。

**変数等のデータが多すぎます**

原因：変数等のデータが多すぎました。

対処：DEF型命令による変数の宣言は128個を超えないようにしてください。

**メモリが足りません**

原因：使用可能なメモリ量が足りませんでした。（コンパイラを起動するために必要なメモリは最低２MBです。）

対処：・RAMを増設してください。
　　　・RAMディスクを設定している場合は、削除してください。

**文字列の長さが許される範囲を超えています**

原因：・プログラムミスまたはオペレーションミスです。
・文字変数に256 文字以上代入しようとしました。

対処：・文字変数などの演算結果が256 文字以上にならないようにしてください。
・キーまたは入出力装置より１つの文字変数に256 文字以上代入しないようにしてください。

**文字列（変数）は使用できません**

原因：ここでの文字列（変数）の使用は許可されません。

**ユーザ定義関数名が正しくありません。**

原因：DEF FN命令でFNとパラメタリストの間に関数名がありません。

**読み込むデータがありません**

原因：ソースプログラム中にBASICの正しい文がありません。

対処：コンパイルしようとしたファイルが、BASICエディタを使ってASCII形式で保存されたソースプログラムファイルでない可能性があります。ファイルの内容を確認してください。

**ラベル: 定義されていないラベルが参照されました**

原因：プログラム中で定義していないラベル名を参照しようとしました。

対処：ラベル名が正しいか、または参照したラベル名が定義されているかを確認してください。

**ラベル名が重複して定義されています**

原因：同じラベル名が重複定義されています。

対処：定義するラベル名は、必ず異なったものを使用してください。

# 2. 警告メッセージ

**警告メッセージ**

プログラムミスの可能性があることを通知します。実行形式ファイルは作成されますが、実行する前にメッセージの内容を確認してください。

コンパイル時に表示されるエラーには次のものがあります。

**COMMON指定とALL指定があります**

意味：CHAIN, ALLとCOMMONが両方存在しています。

**XXX: 変数が参照されていません**

意味：宣言されている変数がどこにも使用されていません。

**XXX: 関数二重定義されています**

意味：同じ名前のユーザー定義関数が複数定義されています。

**XXX: 同名の変数が定義されています。**

意味：このユーザー定義関数と同じ名前の変数が既に定義されています。
（後から指定されたユーザー定義関数が優先されます。）

# 3. 実行時のメッセージ

## 実行時のエラー

コンパイラでは、ほとんどのエラーはコンパイル時に発見されますが、一部、実行時に発見されるものもあります。

実行時にエラーが発見されたときは、エラーウィンドウが表示されます。

- エラーウィンドウには、エラーの起きた行番号と、エラー番号が表示されます。エラー番号が示すエラーの原因と対処はインタプリタのときと同じです。
  （☞「F-BASIC386 V2.1 リファレンス」「F-BASIC386　エラーメッセージ一覧」）
- ［確認］を左クリックするとBASICエディタに戻ります。
- コンパイル時の指定で、行番号情報を“持たない”にしたときは、ガイドウィンドウにエラーの起きた行番号は表示されません。

3

# 付　録

F-BASIC386ガイドを便利にご利用いただけるようにエディタ機能一覧や各種の説明を掲載しています。
『エディタ機能一覧』は、エディタで作業中に、使える機能を調べたりするのに便利なように、F-BASIC386のエディタで使える機能が表で一覧できます。
『コンソールシステムでのディレクトリの移動方法』と、『「コンパイラ」と「リンカ」の単独起動』は、F-BASIC386コンパイラでコンソールシステムを使う際に知っていると便利な知識を説明しています。
『F-BASICプログラムの実行』は、FM7,77,77AV系のプログラムを、F-BASIC386で実行するときに必要な手順を説明しています。

# 1. F-BASIC386エディタ機能一覧

ボタン機能一覧：マウスカーソルを合わせて左クリックすると機能します。

| ボタン | 機　　能 |
|---|---|
| 表示切り替えﾎﾞﾀﾝ | マウスの左ボタンを押している間、インタプリタ画面を表示する |
| HELPﾎﾞﾀﾝ | オンラインマニュアルを表示する |
| EXITﾎﾞﾀﾝ | F-BASIC386を終了し、TownsMENU へ戻る |

メニューバー機能一覧：メニューバーのサブメニューから目的の機能を左クリックして使います。

| | ﾌﾟﾙﾀﾞｳﾝﾒﾆｭｰ | | 機　　能 |
|---|---|---|---|
| | aboutFB386 | | F-BASIC386のﾊﾞｰｼﾞｮﾝ 、ﾚﾍﾞﾙ を表示する |
| | Ｈｅｌｐ | | オンラインマニュアルを表示する |
| | 終了 | | F-BASIC386を終了し、TownsMENU へ戻る |
| ファイル | 読み込み | 読み込み | CD-ROMやﾌﾛｯﾋﾟｨﾃﾞｨｽｸ、ﾊｰﾄﾞﾃﾞｨｽｸ のﾌｧｲﾙをｴﾃﾞｨﾀに読み込む |
| | | 実行 | CD-ROMやﾌﾛｯﾋﾟｨﾃﾞｨｽｸ、ﾊｰﾄﾞﾃﾞｨｽｸ のﾌｧｲﾙをｴﾃﾞｨﾀに読み込んで実行する |
| | | マージ | CD-ROMやﾌﾛｯﾋﾟｨﾃﾞｨｽｸ、ﾊｰﾄﾞﾃﾞｨｽｸのﾌｧｲﾙをｴﾃﾞｨﾀに読み込んでマージする |
| | 実行 | | CD-ROMやﾌﾛｯﾋﾟｨﾃﾞｨｽｸ 、ﾊｰﾄﾞﾃﾞｨｽｸ のﾌｧｲﾙをｴﾃﾞｨﾀに読み込んで実行する |
| | 保存 | バイナリ | ｴﾃﾞｨﾀのﾌﾟﾛｸﾞﾗﾑ をﾌﾛｯﾋﾟｨﾃﾞｨｽｸ 、ﾊｰﾄﾞﾃﾞｨｽｸ にﾊﾞｲﾅﾘ形式で保存する |
| | | アスキー | ｴﾃﾞｨﾀのﾌﾟﾛｸﾞﾗﾑ をﾌﾛｯﾋﾟｨﾃﾞｨｽｸ 、ﾊｰﾄﾞﾃﾞｨｽｸ にｱｽｷｰ形式で保存する |
| | ﾃﾞｨﾚｸﾄﾘ作成 | | ﾌﾛｯﾋﾟｨﾃﾞｨｽｸ 、ﾊｰﾄﾞﾃﾞｨｽｸ にﾃﾞｨﾚｸﾄﾘを作成する |
| | 名前変更 | | ﾌｧｲﾙの名前を変更する |
| | 削除 | | 不要になったﾌｧｲﾙをﾌﾛｯﾋﾟｨﾃﾞｨｽｸ あるいはﾊｰﾄﾞﾃﾞｨｽｸ から削除する |
| 編集 | 一行UNDO | | 間違って削除したり書き替えてしまった行の内容を元に戻す(ｶｰｿﾙ行のみ) |
| | 一行挿入 | | 行と行の間に何も書かれていない新しい行を挿入する |
| | 検索 | | ﾌﾟﾛｸﾞﾗﾑ の中から指定した文字列を探し出す |
| | 置換 | | ﾌﾟﾛｸﾞﾗﾑ の中から指定した文字列を探し出し、新しい文字列に書き替える |
| | カット | | ﾌﾟﾛｸﾞﾗﾑ 文の一部を切り取って削除し、切り取った内容はﾊﾞｯﾌｧに保存する |
| | コピー | | ﾌﾟﾛｸﾞﾗﾑ 文の一部を複写し、ﾊﾞｯﾌｧに保存する |
| | ペースト | | ﾊﾞｯﾌｧの内容を指定した位置に挿入する |
| | ﾎﾟｹｯﾄｲﾝ | | ﾌﾟﾛｸﾞﾗﾑ 文の一部を複写し、ﾎﾟｹｯﾄに保存する |
| | ﾎﾟｹｯﾄｱｳﾄ | | ﾎﾟｹｯﾄに保存されているﾃｷｽﾄを指定した位置に挿入する |

# 1　F-BASIC386エディタ機能一覧

メニューバー機能一覧：メニューバーのサブメニューから目的の機能を左クリックして使います。

| | ﾌﾟﾙﾀﾞｳﾝﾒﾆｭｰ | 機　　能 |
|---|---|---|
| 表示 | 再表示 | 編集画面の表示内容を再度表示し直す |
| | 指定行表示 | ﾌﾟﾛｸﾞﾗﾑ を指定した行から表示する |
| | Mark | ﾌﾟﾛｸﾞﾗﾑ の中のある行を指定して、その行にｶｰｿﾙを飛ばす |
| | 12/16dot font | 画面に表示する文字の大きさを変更する |
| 実行 | Run | ｴﾃﾞｨﾀ内のﾌﾟﾛｸﾞﾗﾑ を１行目から実行する |
| | Run 行番号 | ｴﾃﾞｨﾀ内のﾌﾟﾛｸﾞﾗﾑ を指定した行から実行する |
| | Run(Tron) | 実行中の行番号を表示しながらﾌﾟﾛｸﾞﾗﾑ を実行する |
| | Continue | ［STOP］キーで中断したﾌﾟﾛｸﾞﾗﾑ を、中断した行番号から再度実行する |
| | 一行実行 | ﾌﾟﾛｸﾞﾗﾑ 作成中にｳｨﾝﾄﾞｳ に命令を入力して直接実行する |
| その他 | Auto | ﾌﾟﾛｸﾞﾗﾑ 作成時に行番号を自動的に発生させる |
| | Renum | ﾌﾟﾛｸﾞﾗﾑ の行番号を付け直す |
| | 印刷 | ｴﾃﾞｨﾀ上のﾌﾟﾛｸﾞﾗﾑ をﾌﾟﾘﾝﾀで印刷する |
| | ﾌｫﾝﾄ名入力 | DEF FONT命令で指定するﾌｫﾝﾄ名を選択する |
| | 新規作成 | ﾒﾓﾘ 上のﾌﾟﾛｸﾞﾗﾑ をすべて消去する |

ファイルウィンドウの操作：ファイルの指定が必要な機能を実行すると、画面にファイルウィンドウが表示されます。ウィンドウ内の表示を左クリックすると機能します。

| 画面表示 | 機　　能 |
|---|---|
| ◁(ﾄﾞﾗｲﾌﾞﾁｪﾝｼﾞｱｲｺﾝ) | 選択されているﾄﾞﾗｲﾌﾞ番号を降順で変える |
| ▷(ﾄﾞﾗｲﾌﾞﾁｪﾝｼﾞｱｲｺﾝ) | 選択されているﾄﾞﾗｲﾌﾞ番号を昇順で変える |
| ｶﾚﾝﾄﾄﾞﾗｲﾌﾞ | 選択されているﾄﾞﾗｲﾌﾞ番号を表示する |
| ｶﾚﾝﾄﾃﾞｨﾚｸﾄﾘ | 選択されているﾃﾞｨﾚｸﾄﾘ名を表示する |
| SHOW ALL | 設定されているすべてのﾄﾞﾗｲﾌﾞを一覧表示する |
| ﾃﾞｨｽｸの残り容量 | ｶﾚﾝﾄﾄﾞﾗｲﾌﾞの残り容量をキロバイトで表示 |
| ［ﾃﾞｨﾚｸﾄﾘ移動］ | 親ﾃﾞｨﾚｸﾄﾘとｻﾌﾞﾃﾞｨﾚｸﾄﾘの間でｶﾚﾝﾄﾃﾞｨﾚｸﾄﾘを変更する |
| ﾃﾞｨﾚｸﾄﾘ移動ﾎﾞﾀﾝ | ｻﾌﾞﾃﾞｨﾚｸﾄﾘから親ﾃﾞｨﾚｸﾄﾘへｶﾚﾝﾄﾃﾞｨﾚｸﾄﾘを変更する |
| ファイル名リスト | ﾌｧｲﾙおよびｻﾌﾞﾃﾞｨﾚｸﾄﾘの表示 |
| ｽｸﾛｰﾙﾊﾞｰ | 任意の箇所を左クリックして、ﾌｧｲﾙﾒﾆｭｰの表示をｽｸﾛｰﾙ する |
| ｲﾝｼﾞｹｰﾀ | マウスの左ボタンでドラッグさせて、ﾌｧｲﾙﾒﾆｭｰの表示を任意にｽｸﾛｰﾙ する |

## 1　F-BASIC386エディタ機能一覧

ファイルウィンドウの操作：ファイルの指定が必要な機能を実行すると、画面にファイルウィンドウが表示されます。ウィンドウ内の表示を左クリックすると機能します。

| 画面表示 | 機　　能 |
|---|---|
| ▲（ｽｸﾛｰﾙﾎﾞﾀﾝ） | ﾌｧｲﾙﾒﾆｭｰの表示を上へｽｸﾛｰﾙ する |
| ▼（ｽｸﾛｰﾙﾎﾞﾀﾝ） | ﾌｧｲﾙﾒﾆｭｰの表示を下へｽｸﾛｰﾙ する |
| 選択ファイル名の枠 | 選択されているﾌｧｲﾙ名の表示およびﾌｧｲﾙ名の入力 |
| ［実行］または⏎ | 選択内容と機能の実行 |
| ［取消］またはESC | 選択内容の取消（ﾌｧｲﾙｳｨﾝﾄﾞｳの消去） |

キー機能一覧：エディタ画面の表示中にキーボードのキーを直接操作して機能を使えます。

| | キー操作 | 機　　能 |
|---|---|---|
| カーソル移動 | ↑ | カーソルを上へ移動する |
| | ↓ | カーソルを下へ移動する |
| | ←またはCTRL＋S | カーソルを左へ移動する |
| | →またはCTRL＋D | カーソルを右へ移動する |
| | CTRL＋A | ｶｰｿﾙを単語の先頭または直前の単語の先頭へ移動する |
| | CTRL＋F | ｶｰｿﾙを次の単語の先頭へ移動する |
| | ⏎またはCTRL＋M | 入力を終了し、ｶｰｿﾙを次の行の先頭へ移動する |
| 削除 | ［後退］（JIS は⇦） | カーソルの直前の文字を１文字削除する |
| | ［削除］ | カーソルの右側の文字を１文字削除する |
| | SHIFT＋［削除］または CTRL＋E | ｶｰｿﾙの右側から行末までの文字をすべて削除する |
| サブメニューの表示 | PF 1 | ［ファイル］のサブメニューを表示する |
| | PF 2 | ［編集］のサブメニューを表示する |
| | PF 3 | ［表示］のサブメニューを表示する |
| | PF 4 | ［実行］のサブメニューを表示する |
| | PF 5 | ［その他］のサブメニューを表示する |
| ﾌｧｲﾙ | SHIFT＋CTRL＋L | ［読み込み］を選択する |
| | SHIFT＋CTRL＋S | ［保存］を選択する |

# 1　F-BASIC386エディタ機能一覧

キー機能一覧：エディタ画面の表示中にキーボードのキーを直接操作して機能を使えます。

| | キー操作 | 機　　能 |
|---|---|---|
| プログラムの編集 | CTRL＋U | 行の内容を復元する　（［一行UNDO］） |
| | CTRL＋Qまたは SHIFT＋［挿入］ | 行を挿入する　（［一行挿入］） |
| | CTRL＋L | ［検索］を選択する　（［検索］） |
| | CTRL＋P | 上方向に検索する |
| | CTRL＋N | 下方向に検索する |
| | SHIFT＋CTRL＋X | ﾌﾟﾛｸﾞﾗﾑ を専用メモリに移動する（［カット］） |
| | SHIFT＋CTRL＋C | ﾌﾟﾛｸﾞﾗﾑ を専用メモリに複写する（［コピー］） |
| | SHIFT＋CTRL＋V | 専用メモリの内容を挿入する　（［ペースト］） |
| | CTRL＋←または→ | 一字範囲指定 |
| | SHIFT＋CTRL＋←または→ | 一語範囲指定 |
| | CTRL＋［実行］ | マークを設定する　（［Mark］） |
| 再表示 | CTRL＋Kまたは HOME | ﾌﾟﾛｸﾞﾗﾑ を表示し直す　（［再表示］） |
| 実行 | CTRL＋G | ﾌﾟﾛｸﾞﾗﾑ を実行する　（［Run ］） |
| ボタンの選択 | PF11 | オンラインマニュアルを表示する　（HELPボタン） |
| | PF12 | F-BASIC386を終了し、TownsMENU へ戻る（EXITボタン） |

# 2. コンソールシステムでのディレクトリの移動方法

BASICコンソールシステムでは、MS-DOSの内部コマンドが使用できます。特に、使用頻度の高い“CD”コマンドについて説明します。その他のコマンドについては、MS-DOSのマニュアルをご覧ください。

コンソールシステムでディレクトリを移動するときは、プロンプトに続けて、コマンドと移動先ディレクトリ名を直接入力します。

プロンプトに続けて次の形式で入力し、⏎キーを押してください。

```
cd ディレクトリ名
```

cd：Change Directoryの略で、ディレクトリを移動するためのコマンドです。
ディレクトリ名：移動先のディレクトリ名を入力します。

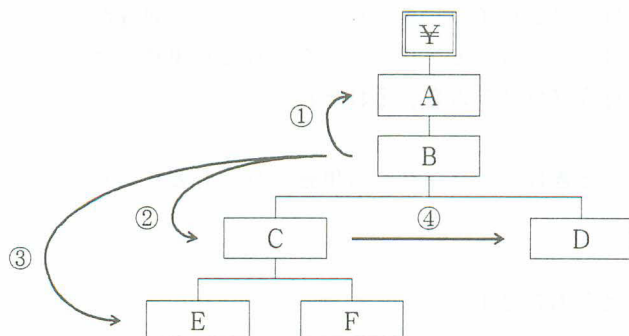
☞ディレクトリを移動すると、プロンプトに移動後のディレクトリが表示されます。

**ディレクトリ名の指定**

ディレクトリ名は次の規則にしたがっ指定します。

- すぐ上、あるいはすぐ下のディレクトリに移動するときは、移動先のディレクトリ名をそのまま入力します。また、親ディレクトリは..で表わされ、「cd ..」と入力して、すぐ上のディレクトリに移動することもできます。
- 2階層以上を一度に移動するときは、どのディレクトリを通って、目的のディレクトリに移動するかを示すパス名を入力します。このとき、ディレクトリとディレクトリの間は¥でつなぎます。
  ☞パス名とは、階層ディレクトリ構造において、特定のディレクトリへの道すじ（どのディレクトリを通るか）を示す文字列です。
- ルートディレクトリは¥で表します。

## 2　コンソールシステムでのディレクトリの移動方法

具体的なディレクトリ名の指定方法を、次のような階層ディレクトリを例に示します。

| ①B→A | cd .. | . . は親ﾃﾞｨﾚｸﾄﾘを示す。　→相対指定※ |
|---|---|---|
| | cd ¥A | ﾙｰﾄ（¥）を起点としてAに移る。　→絶対指定※ |
| ②B→C | cd C | すぐ下のﾃﾞｨﾚｸﾄﾘに移動するときは、そのﾃﾞｨﾚｸﾄﾘ名を直接入力する。　→相対指定※ |
| | cd ¥A¥B¥C | ﾙｰﾄ（¥）を起点とし、A、Bを通ってCに移る。　→絶対指定※ |
| ③B→E | cd C¥E | すぐ下のﾃﾞｨﾚｸﾄﾘCを通ってEに移る。　→相対指定※ |
| | cd ¥A¥B¥C¥E | ﾙｰﾄ（¥）を起点とし、A、B、Cを通ってEに移る。　→絶対指定※ |
| ④C→D | cd ..¥D | 親ﾃﾞｨﾚｸﾄﾘを通ってDに移る。　→相対指定※ |
| | cd ¥A¥B¥D | ﾙｰﾄ（¥）を起点とし、A、Bを通ってDに移る。　→絶対指定※ |
| →ルート | cd ¥ | どのﾃﾞｨﾚｸﾄﾘからでもいきなりﾙｰﾄﾃﾞｨﾚｸﾄﾘ に戻ることができます。 |

※　絶対指定と相対指定

パス名の表し方には、「絶対指定」と「相対指定」があります。

絶対指定：現在のディレクトリとは無関係に、ルートディレクトリからの道すじを指定する方法です。

相対指定：現在のディレクトリを起点とし、親ディレクトリを通って道すじを指定する方法です。

# 3.「コンパイラ」と「リンカ」の単独起動

TownsMENUで[COMPLIER]を実行したとき、あるいは、コンソールシステムで“BC”コマンドを入力してコンパイルしたとき、その内部では「コンパイル」と「リンク」の２つの処理が行われます。
コンソールシステムでは、この「コンパイル」と「リンク」をそれぞれ単独で行うことができます。

- 「コンパイル」、「リンク」を行うプログラムを、それぞれ「コンパイラ」、「リンカ」といいます。
- 「コンパイラ」を単独で使用したときは、「リンク」も単独で行う必要があります。

**「コンパイラ」を単独で起動する**

プロンプトに続けて、次のように入力します。

```
RUN386 (パス名)FB386CP [パラメータ] ソースプログラムファイル名
```

- “FB386CP” にはパス名が必要です。
- ［パラメータ］には、「4.コンパイル時の条件の変更（244ページ）」と同じ文字列が指定できます。［パラメータ］は半角文字の小文字で指定します。
- 「コンパイル」を行うと、ソースプログラムファイルと同じ名前で拡張子が“.OBJ”の、オブジェクトファイルが生成されます。

例）“FB386CP” がＥドライブの ¥BASIC¥BASCOMV21¥ディレクトリにあり、“SAMPLE.BAS”というソースプログラムをコンパイルし、アセンブルするときのもととなるファイルをつくるとき

```
RUN386 E:¥BASIC¥BASCOMV21¥FB386CP -asm SAMPLE
```

**「リンカ」を単独で起動する**

プロンプトに続けて、次のように入力します。

```
RUN386 (パス名)TLINKP @リンクファイル名
```

- “TLINK” にはパス名が必要です。
- リンクファイルはリンカの動作を指定するファイルです。リンクファイルはTownsシステムソフトウェアの「テキスト編集」を使って作成します。
  - リンクファイルの拡張子は、“.LNK”にします。
  - リンクファイルを作成するときは、Townsシステムソフトウェアの「テキスト編集」で、“BC.BAT”を変更し、名前を変更して保存する方法が便利です。

3

リンクファイルは次のような形式で作成します。

```
ｵﾌﾞｼﾞｪｸﾄﾌｧｲﾙ名
-exe 実行ﾌﾟﾛｸﾞﾗﾑﾌｧｲﾙ名
-386 -tc -stack 77824
-lib (ﾊﾟｽ名)baslib.lib
(ﾊﾟｽ名)snd.lib (ﾊﾟｽ名)tbios.lib
(ﾊﾟｽ名)fmcfrb.lib (ﾊﾟｽ 名)cdrfrb.lib
```

「（パス名）」には、<bascom>ディレクトリのパス名を指定します。

例）リンクファイル、オブジェクトファイル、実行プログラムファイルの名前がそれぞれ、“SAMPLE.LNK”、“SAMPLE.OBJ”、“SAMPLE.EXP”で、“TLINKP”がＥドライブの¥BASIC¥BASCOM にあるとき

＜リンクファイル＞

```
SAMPLE
-exe SAMPLE
-386 -tc -stack 77824
-lib E:¥basic¥bascom¥baslib.lib
E:¥basic¥bascom¥snd.lib E:¥basic¥bascom¥tbios.lib
E:¥basic¥bascom¥fmcfrb.lib E:¥basic¥bascom¥cdrfrb.lib
```

＜リンカの起動＞

```
RUN386 E:¥BASIC¥BASCOM¥TLINKP @SAMPLE
```

# 4. F-BASICプログラムの実行

FM7,77,77AV系のF-BASICプログラムを、F-BASIC386で実行するには、ファイルの変換と、非互換命令の変更が必要です。

## F-BASICのファイルを変換する

F-BASICのファイルコンバータFBCNVで、ファイルを変換します。

- F-BASICのファイルはアスキー形式で保存されている必要があります。
- グラフィックエディタやミュージックエディタのデータは変換してもLOAD@命令で読み込めません。

**ひとこと**

● FBCNVでは、F-BASIC386からF-BASICへのファイルの変換も可能です。
ただし、Townsでは2Dのフロッピィディスクへの書込みはできないので、F-BASIC386からF-BASICへ変換する場合は、2DDのフロッピィディスクを用意してください。

**1 F-BASIC386のCD-ROMを起動します。**

TownsMENUが表示されます。

**2 FBCNVを選択します。**

FBCNVのアイテムを選択して実行させると、FBCNVの初期画面が表示されます。

**3 フロッピィディスクをセットします。**

変換元と変換先のフロッピィディスクをセットし、ドライブ番号と変換方向（F-BASIC→TownsOS）を設定します。

**ご 注 意**

● 1.44MBでフォーマットしたフロッピィディスクは、使用できません。

**4 変換元のプログラムファイルを選択します。**

フロッピィディスクに保存されているファイルの一覧が表示されるので、変換するプログラムを選択します。
一度に複数個のファイルを指定できます。

5　実行ボタンを左クリックすると、変換が実行されます。

変換開始時に、変換先のフロッピィディスクに同じファイル名があったときは、再試行、次試行または中止のいずれかを選択します。

再試行：そのファイルを書き替えます。（そのファイルの元のデータは失われます。）
次試行：そのファイルだけ変換しないで次に進みます。
中　止：変換自体を中止します。

「＊」「？」などTownsOSで使えないファイル名の場合は、ファイル名を聞いてきますので、正しいファイル名を入力します。

### 変換したファイルを修正する

F-BASICとF-BASIC386の各命令規約の違い（非互換部分）を、F-BASIC386のエディタをつかって修正します。

● 予約語の前後は、１文字以上の空白または特殊記号で切離します。

```
GOTO200       →  GOTO 200
COLOR2,6      →  COLOR 2,6
DATA1,リンゴ  →  DATA 1, リンゴ
```

● キャラクタ座標の設定を変えます。

```
WIDTH 40,25   →  WIDTH 80,25
```

● 画面モードの定義を変えます。

☝SCREEN@の定義を変更します。SCREEN命令は削除します。

| 画面モード 77AV | | 画面モード Towns |
|---|---|---|
| 0：8/8色 640×200 | → | 0：16/4096色 640×480 |
| 1：4096/4096色 320×200 | → | 1：32768色 パレットなし 320×240 |
| 2：8/8色 640×400 | → 0 | 2：256/16 百万色 640×480 |
| 3：26 万色 パレットなし 320×200 | → 1 | |

● グラフィック画面の座標を合わせます。

☝プログラムの最初でウィンドウを設定して、FM7,77,77AVと同じ座標にします。

400 ラインの場合　：WINDOW (0,0)-(639,399)
200 ラインの場合　：WINDOW (0,0)-(639,199)
4096色の場合　　　：WINDOW (0,0)-(319,199)

● パレット命令を変更します。

☝32768 色モードでは、パレットは使えません。
PALETTE@、COLOR=を別の色指定方法に変えます。

☝16色モードでは、パレットは使えますが、パレット命令が異なります。

PALETTE@ → PALETTE
COLOR= → （16色モード）

● 円の縦横比率を変更します。

　円の縦横比率を指定している場合は、比率を変更する必要があります。

● SOUND 命令は使えません

　PLAY命令で、FM音色の効果音などに変更します。

● PSET命令を変更します。

```
PSET (300,100,2)  →  PSET (300,100),2
```

● GET@命令を変更します。

　配列の大きさを変更します。

　G指定を削除します。

● CONSOLE 命令を変更します。

```
CONSOLE 0,25,0,0,0  →  CONSOLE 0,25,0
```

● 以下の機能は、F-BASIC386にはありません。

- グラフィックキャラクタの機能
  ファイル名、表などで使っていたら、変更します。
- LINE命令で、文字を使って線や箱を描く機能
- GET@/PUT@ で文字（キャラクタ座標の）をget/put する機能

# 索引

## ファイル名



## サブディレクトリ



## 数字・アルファベット

# 索　引

# 索　引



## あ行

# 索　引

## さ行

## 索　引

# 索　引

# 索引

## ま行



## や行



## ら行



## わ行